UNIVERSITAT ROVIRA I VIRGILI (URV) Y UNIVERSITAT OBERTA DE CATALUNYA (UOC)

MÁSTER UNIVERSITARIO EN INGENIERÍA COMPUTACIONAL Y MATEMÁTICA

TRABAJO FINAL DE MÁSTER

ÁREA: Matemática aplicada

Simulación de muelles.

Realización de un sistema de estimación de la constante elástica de un muelle helicoidal mediante el método de los elementos finitos.

Autor: Juan Pasamar Escudero

Tutor: Gerard Fortuny Anguera y Josep Maria López Besora

Fecha de Entrega: 05/07/2023

El /a Dr./Dra. Josep Maria Lópes Besora y Gerard Fortuny Anguera, certifica que el/la estudiante Juan pasamar Escudero ha elaborado el trabajo bajo su tutoría y autoriza la presentación de esta memoria para la su evaluación.

Firma del director/a:

Josep Maria López Besora

Gerard Fortuny Anguera

Licencias alternativas
(elegir alguna de las siguientes y sustituir la de la página anterior)

A) Creative Commons:

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento 3.0 España de Creative Commons

B) GNU Free Documentation License (GNU FDL)
Copyright © AÑO TU-NOMBRE.
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU Free Documentation License".

C) Copyright
© (el autor/a)
Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

FICHA DEL TRABAJO FINAL DE MASTER

Título:	**Simulación de muelles.**
Nombre del autor:	Juan Pasamar Escudero.
Nombre del tutor/a:	Gerard Fortuny Anguera Josep Maria López Besora
Fecha (mm/aaaa)**:**	07/2023
Titulación:	Máster en Ingeniería Computacional y Matemática
Área:	Área del TFM
Idioma:	**Español**
Palabras clave:	*Muelle, FEM, Elástica*

Dedicatoria/Cita

A Elena, Marta y Lucía.

Agradecimientos

A Gerard Fortuny Anguera y Josep Maria López Besora, por ofrecerme un tema tan interesante en el que contribuir y por guiarme hasta su final.

Resumen

Este trabajo presenta una herramienta para determinar la constante elástica de un muelle helicoidal utilizando modelos de elementos finitos en 1D y 3D. El objetivo es desarrollar una aplicación que siga una metodología que pueda calcular la constante elástica en función de las propiedades geométricas y mecánicas del muelle. El estudio se centra en muelles helicoidales con secciones circulares.

La metodología implica la creación de la geometría del muelle y su discretización en elementos finitos utilizando el software Gmsh. Se consideran dos enfoques: un modelo en 1D y un modelo en 3D. El modelo en 1D representa el muelle como una serie de elementos viga, mientras que el modelo en 3D lo representa elementos que forman un volumen sólido.

El software Code_Aster se utiliza para resolver el problema mecánico y obtener los resultados. Las simulaciones se validan mediante la comparación de las constantes elásticas calculadas con los valores proporcionados por los fabricantes de los muelles. El análisis incluye variaciones en el tamaño de la malla y considera diferentes modelos de muelles para evaluar la convergencia y precisión de los resultados.

Los hallazgos demuestran que tanto el modelo en 1D como el modelo en 3D convergen hacia los valores teóricos de la constante elástica a medida que disminuye el tamaño de la malla. Sin embargo, el modelo en 3D proporciona resultados más precisos a costa de un mayor tiempo de cálculo. El modelo en 1D ofrece un buen equilibrio entre precisión y eficiencia.

El trabajo resalta la importancia del tamaño de la malla para obtener resultados precisos e identifica un equilibrio entre precisión y coste computacional. También se discuten los aspectos críticos de la metodología y se presentan futuras líneas de investigación.

Este estudio proporciona una herramienta para determinar la constante elástica de un muelle helicoidal mediante análisis de elementos finitos. Ofrece conocimientos sobre la precisión, eficiencia y aplicabilidad de los modelos, sentando las bases para futuros desarrollos que amplíen y mejores las capacidades de las aplicaciones obtenidas.

Palabras clave: *muelle, helicoidal, constante elástica, elementos finitos, modelado 1D, modelado 3D, Gmsh, Code_Aster, tamaño de malla, precisión, eficiencia, validación, geometría, simulación, análisis, convergencia, metodología, optimización.*

Abstract

This work presents a tool for determining the spring constant of a helical spring using 1D and 3D finite element models. The aim is to develop an application that follows a methodology capable of calculating the spring constant based on the spring's geometric and mechanical properties. The study focuses on helical springs with circular cross-sections.

The methodology involves creating the spring's geometry and discretizing it into finite elements using the Gmsh software. Two approaches are considered: a 1D model and a 3D model. The 1D model represents the spring as a series of beam elements, while the 3D model represents it as elements that form a solid volume.

The Code_Aster software is used to solve the mechanical problem and obtain the results. Simulations are validated by comparing the calculated spring constants with the values provided by the spring manufacturers. The analysis includes variations in mesh size and considers different spring models to evaluate the convergence and accuracy of the results.

The findings show that both the 1D and 3D models converge to the theoretical values of the spring constant as the mesh size decreases. However, the 3D model provides more accurate results at the cost of increased computation time. The 1D model offers a good balance between accuracy and efficiency.

The work highlights the importance of mesh size for obtaining accurate results and identifies a balance between accuracy and computational cost. The critical aspects of the methodology are also discussed, and future lines of research are presented.

This study provides a tool for determining the spring constant of a helical spring through finite element analysis. It offers insights into the accuracy, efficiency, and applicability of the models, laying the foundation for future developments that expand and improve the applications' capabilities.

Keywords: spring, helical, spring constant, finite elements, 1D modeling, 3D modeling, Gmsh, Code_Aster, mesh size, accuracy, efficiency, validation, geometry, simulation, analysis, convergence, methodology, optimization.

Índice

Índice de figuras

1.Introducción

1.1.Contexto y justificación del Trabajo

La constante elástica es un parámetro de uso común cuando se trabaja con muelles. Este parámetro es el que relaciona la fuerza ejercida por un muelle con la deflexión a la que este está sometido. De forma habitual, esta constante es proporcionada por el fabricante del muelle, quien la estima utilizando un método analítico.

Desde este trabajo, se propone un sistema que permite obtener esta constante utilizando diferente metodología para las características de geometría y material dadas para cada muelle. Esto permite tener un control más preciso sobre las propiedades del muelle y su comportamiento mecánico, lo cual puede ser crucial en aplicaciones donde se requiere un diseño específico.

En este trabajo, se propone resolver el problema de la obtención de la constante elástica mediante el método de los elementos finitos (FEM). Este método es una técnica numérica utilizada para aproximar soluciones a problemas complejos de ingeniería y ciencias aplicadas. En el caso de este trabajo, se utilizará el FEM para modelar la geometría del muelle, su comportamiento y obtener la constante elástica.

La aportación realizada consiste en desarrollar un programa que utilice el FEM para obtener la constante elástica del muelle. Este programa tomará como entrada los datos geométricos del muelle, las propiedades mecánicas del material y aplicará el método de los elementos finitos (creación de la geometría, mallado, resolución y postprocesado) para calcular la constante elástica. El resultado deseado es obtener una estimación precisa de la constante elástica del muelle, personalizada para las características específicas del muelle en cuestión. Esto proporcionaría una herramienta útil para el diseño y análisis de sistemas que involucren el uso de muelles.

1.2.Objetivos del Trabajo

El trabajo tiene como objetivo principal desarrollar un sistema que permita la obtención de la constante elástica de un muelle utilizando el método de los elementos finitos (FEM). Los objetivos específicos son los siguientes:

1. Investigar y comprender el método de los elementos finitos y su aplicación en el análisis de estructuras elásticas, en particular en el caso de muelles.

2. Desarrollar una rutina que recopile los datos geométricos del muelle, genere la geometría del muelle y cree una malla para ser utilizada en el análisis posterior.

3. Desarrollar una herramienta que implemente el método de los elementos finitos para modelar el comportamiento elástico del muelle y que calcule las deformaciones producidas en el muelle frente a una serie de restricciones y fuerzas aplicadas.

4. Validar el programa mediante comparaciones con resultados experimentales o soluciones analíticas conocidas.

5. Realizar pruebas y análisis de sensibilidad para evaluar la influencia de diferentes parámetros en la constante elástica obtenida.

6. Evaluar las ventajas y limitaciones de los enfoques propuestos en comparación con los métodos tradicionales de obtención de la constante elástica.

7. Identificar posibles áreas de mejora o futuras investigaciones relacionadas con el tema.

8. Documentar los resultados obtenidos, incluyendo los procedimientos utilizados, las conclusiones obtenidas y las posibles limitaciones de los métodos propuestos.

Al cumplir estos objetivos, se espera obtener un sistema que proporcione una estimación precisa de la constante elástica de un muelle utilizando el método de los elementos finitos..

1.3.Enfoque y método seguido

El enfoque seguido en este trabajo es el desarrollo de un sistema que automatice el proceso en la obtención de la constante elástica. La estrategia elegida es desarrollar un producto a partir de librerías y software disponible ya existente que implemente la generación de la geometría, el mallado de la misma y análisis mecánico.

Esta estrategia permite abordar de manera precisa y personalizada la obtención de la constante elástica para cada muelle. Al desarrollar un programa específico para esta tarea, se pueden considerar las características geométricas del muelle, las propiedades mecánicas del material y otras variables relevantes para obtener una estimación precisa de la constante elástica. Además, el uso de este método proporciona una base sólida y confiable para realizar el análisis.

Además, al desarrollar el programa informático, se logra una herramienta práctica y fácil de usar para el diseño y análisis de sistemas que involucren el uso de muelles. Las herramientas de este tipo permiten a ingenieros y diseñadores obtener rápidamente estimaciones de las propiedades de los elementos mecánicos y evaluar diferentes configuraciones para optimizar el diseño y garantizar un comportamiento adecuado.

En resumen, el enfoque elegido de utilizar librerías existentes para aplicar este tipo de análisis es una estrategia que permite obtener estimaciones precisas y personalizadas, proporcionando una herramienta útil para el diseño y análisis en sistemas que los utilizan.

1.4.Planificación del Trabajo

Para realizar este trabajo, se ha utilizado los paquetes de software GMSH y code_aster. Ambos paquetes están registrados con licencias de software libre que permiten su uso.
El desarrollo del software y la obtención de resultados se ha realizado con un ordenador portátil como estación de trabajo.
En la siguiente figura, se encuentra la temporalización que se ha seguido. Se debe tener en cuenta que en los primeros meses, las horas invertidas en el desarrollo han estado mucho mas espaciadas que en el resto del proyecto, por lo que aunque temporalmente parezca que las primeras etapas del trabajo la inversión de horas ha sido menor, en términos generales se ha invertido mayor cantidad de horas en la parte final del trabajo, conforme se ha tenido mas disponibilidad horaria para dedicarle.

	Febrero						Marzo					Abril					Mayo				Junio					Julio
TÍTULO DE LA TAREA	**Documentación sobre el tema y panteamiento de soluciones.**											**Desarrollo de aplicación.**									**Obtención de resultados.**				**Maquetado y generación de informe.**			
	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W18	W19	W20	W21	W23	W24	W25	W26	W27	W28		W29	W30	W31	W32
Establecimiento de objetivos																												
Investigación de la literatura existente	x	x	x	x	x	x	x																					
Análisis y resumen de hallazgos de la investigación			x	x	x	x	x	x	x																			
Identificación de posibles soluciones y planteamiento del enfoque							x	x	x	x	x																	
Desarrollo de la aplicación																												
Diseño de la arquitectura de la aplicación												x	x	x														
Creación de la aplicación														x	x													
Pruebas iniciales y correcciones															x	x												
Modificaciones y ajustes de la aplicación																x	x	x	x	x								
Obtención de resultados																												
Pruebas de la aplicación																					x	x						
Recolección y análisis de resultados																						x	x					
Interpretación de resultados																							x					
Preparación de informes preliminares																							x	x				
Maquetado y generación de informe																												
Creación del primer borrador del informe																									x			
Revisión y edición del borrador																									x	x	x	
Preparación y finalización del informe final																										x	x	x

Figura 1: Diagrama de Gantt del trabajo.

1.5.Breve sumario de productos obtenidos

Tal como se detalla en capítulos posteriores, con la realización de este trabajo se han obtenido diferentes herramientas como se indica a continuación:

1. Una rutina para la generación de la geometría y la malla de un muelle en 3D con GMSH.

2. Un lanzador para la generación de los archivos de configuración para lanzar los análisis 3D con codeaster.

3. Una herramienta para la generación de la geometría y la malla de un muelle en 1D con GMSH.
4. Un lanzador para la generación de los archivos de configuración para lanzar los análisis 1D con codeaster.
5. Una función que nos encadena la secuencia de trabajo para obtener la constante elástica en 1D.
6. Una función que nos encadena la secuencia de trabajo para obtener la constante elástica en 3D.
7. Un script que abre un archivo .csv, copia los datos existentes en el y lanza los casos de estudios que estén sin resolver. Para terminar actualiza el contenido del archivo con los resultados obtenidos.

1.6.Breve descripción de los otros capítulos de la memoria

A continuación, se proporciona un breve resumen de cada uno de los capítulos incluidos en esta memoria, describiendo sus contenidos y cómo se relacionan entre sí y con el trabajo en su conjunto.

- Capítulo 2: Introducción: En este capítulo se proporciona una visión general del trabajo y se presenta el objetivo principal. Se discuten las motivaciones, la relevancia y la estructura general del documento.
- Capítulo 3: Metodología: En este capítulo se describe en detalle la metodología utilizada en el trabajo. Se presenta el procedimiento de simulación, que involucra la creación de la geometría del muelle y su discretización en elementos finitos. Se mencionan los software utilizados, como Gmsh y Code_Aster, y se explican sus principales características y funcionalidades. También se aborda la generación de la malla y se discute el tamaño adecuado de la misma para obtener resultados precisos. Se describen los modelos de simulación utilizados, tanto en 1D como en 3D, y se mencionan los parámetros y variables de entrada considerados en el análisis.
- Capítulo 4: Resultados y Análisis: En este capítulo se presentan los resultados obtenidos a partir de las simulaciones y análisis realizados. Se describen los casos de estudio realizados utilizando diferentes modelos de muelles y tamaños de malla. Se analizan los valores de la constante elástica obtenidos y se comparan con los valores teóricos y los proporcionados por los fabricantes de los muelles. También se estudian los tiempos de cálculo y se discute la eficiencia computacional de los modelos en 1D y 3D. Se realiza un análisis crítico de los resultados y se presentan gráficas y conclusiones relevantes.
- Capítulo 5: Conclusiones: En este capítulo se presentan las conclusiones del trabajo. Se discuten las lecciones aprendidas, se reflexiona sobre el logro de los objetivos planteados inicialmente y

se analiza el seguimiento de la planificación y metodología a lo largo del trabajo. Se presentan líneas de trabajo futuro que no se han explorado en este trabajo y quedan pendientes.

Juntos, estos capítulos forman una evaluación completa del uso del MEF para la simulación de muelles, desde los fundamentos teóricos hasta las aplicaciones prácticas, pasando por el estado actual de la investigación en el campo. Cada capítulo se basa en el anterior, resultando en una presentación coherente y completa del tema.

2.Marco teórico
2.1.Método de los elementos finitos

Para obtener los resultados de este trabajo, se necesita resolver un problema mecánico en un sistema continuo. Los sistemas continuos son elementos geométricos complejos que son difíciles de resolver por métodos analíticos. El método de los elementos finitos aproxima el resultado de las ecuaciones diferenciales del medio continuo a través de ecuaciones algebraicas con un número finito de incógnitas.

Aunque en este caso la solución analítica de un muelle helicoidal es conocida (siempre que consideremos el material con una elasticidad lineal), el trabajo realizado resuelve el problema separando la geometría y convirtiendo el sistema continuo en un número finito de elementos discretos. Esta discretización del sistema es lo que llamamos el método de los elementos finitos, el cual nos permite obtener una solución aproximada a los problemas de ecuaciones diferenciales aplicadas a sistemas continuos.

2.1.1.Principios básicos.

Al dividir la geometría del elemento a analizar en subdivisiones creamos una serie de elementos. Estos elementos, se conectan entre si por unos puntos a los que llamamos nodos.

Hay diferentes tipos de elementos según su forma y el número de nodos que lo componen. Además, los nodos pueden estar únicamente en los vértices o a lo largo de las aristas.

Como se indica en Celigüeta [8], los principales tipos de elementos que nos encontramos cuando trabajamos con elementos finitos son:

- Elementos unidimensionales.

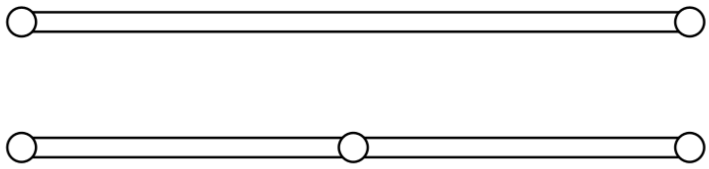

Figura 2: Elementos 1D

- Elementos bidimensionales.

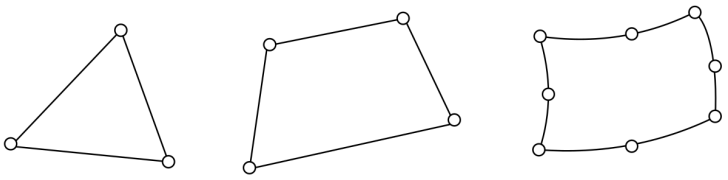

Figura 3: Elementos 2D

- Elementos tridimensionales.

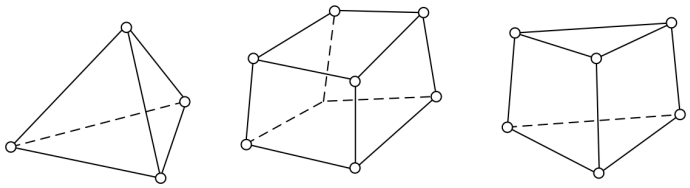

Figura 4: Elementos 3D

La hipótesis de discretización utilizada en el MEF se basa en lo siguiente:

- El elemento continuo se divide en un número determinado de elementos de formas geométricas sencillas, contiguos entre si.
- Estos elementos están conectados entre si por medio de nodos.
- Según el tipo de análisis, la incógnita a resolver será diferente. Para el caso que abarca este proyecto, la incógnita a resolver es la de los desplazamientos en los nodos.
- Cada tipo de elemento tiene una función de forma que determina los desplazamientos de los nudos que lo forma.
- Las funciones de interpolación y los desplazamientos nodales determinan el estado de las deformaciones unitarias del elemento.
- Cada elemento tiene un sistema de fuerzas en los nudos que equilibran las tensiones del elemento con las fuerzas exteriores que actúan sobre él.

2.1.2.Ventajas y limitaciones.

La solución obtenida a través de este método, es una solución aproximada. Como toda solución aproximada, esta presenta ventajas y limitaciones que es conveniente explicar para poder valorar la validez de los modelos desarrollados.

Según Padron [6], las principales ventajas que tiene el MEF son:

- Se puede realizar simulaciones a geometrías y sistemas muy complejos.
- Podemos trabajar con multitud de tipos de excitación, como por ejemplo: cargas puntuales, distribuidas en superficie, a lo largo de aristas, cargas estáticas, cargas dinámicas.
- Con él podemos resolver multitud de problemas. Los mas habituales son casos estáticos y dinámicos de sólidos, transferencias de calor, campos electromagnéticos y mecánica de fluidos.

Sus principales limitaciones son:

- Son soluciones aproximadas, donde de forma general, obtenemos soluciones mas precisas según refinamos la malla.

- Las soluciones obtenidas pueden contener errores inherentes al proceso de cálculo como resultado de la acumulación de errores del sistema de resolución numérico.
- Cuanto mas fina sea la malla, mas preciso es el resultado, pero también aumenta de forma exponencial los tiempos de cálculo.

2.2.Teoría de muelles y conceptos básicos

Podemos definir un muelle como un cuerpo elástico cuya función principal es deformarse ante la presión de una fuerza externa, pero que al desaparecer dicha fuerza externa recobrará su forma original.

Casi todos los elementos mecánicos son elásticos y se deforman ante la acción de una fuerza, pero solo consideraremos muelles aquellos que tengan como función principal la deformación ante la aparición de la fuerza externa.

Como elementos mecánicos tienen multitud de funciones, siendo su uso muy extendido, pudiendo destacar su uso como:

- Absorber energía y reducir los efectos de choques e impactos.
- Aplicar una fuerza o par definido.
- Hacer funciones de soporte de masas.
- Aislar vibraciones.
- Indicar o controlar fuerzas o pares.
- Para ejercer de guía o de pivote elástico.

Según la naturaleza de las cargas a las que están sometidos los muelles, podemos encontrar dos tipos de cargas fundamentales. Estas son las cargas estáticas y las cargas dinámicas.

- Carga Estática: Esta carga se aplica de manera gradual hasta que alcanza un valor constante. Una vez que se llega a este punto, no varía con el tiempo. En el caso de un muelle, una carga estática puede ser una fuerza que lo estira o comprime hasta una longitud determinada y luego se mantiene en esa posición, o un par torsor constante en el caso de muelles de torsión.
- Carga Dinámica: A diferencia de las cargas estáticas, las cargas dinámicas cambian con el tiempo. Esto puede incluir cambios cíclicos, como los que se encuentran en un sistema de suspensión de automóvil, o variaciones más aleatorias, como las cargas que recibe un muelle de un tren al pasar por distintas partes de la vía.

Los muelles, con su diversidad de formas y tamaños, se diseñan específicamente para manejar distintos tipos de carga. Cada muelle se optimiza para permitir una deformación elástica apropiada frente a la carga para la cual fue creado. Sin embargo, estos pueden no funcionar eficientemente al enfrentar fuerzas o torsiones aplicadas en direcciones y sentidos no contemplados en su diseño original.

Según su forma constructiva, los muelles se pueden clasificar en varios tipos:

- Muelles Helicoidales: Son el tipo de muelle más comúnmente utilizado y se fabrican enrollando un alambre o barra de acero alrededor de un eje. Se pueden subdividir en:
  - Muelles Helicoidales de Compresión: Diseñados para resistir la compresión, de modo que la carga causa que el resorte se acorte.
  - Muelles Helicoidales de Tracción: Diseñados para resistir la elongación, por lo que la carga provoca que el resorte se estire. Este tipo de muelle generalmente tiene ganchos o lazos en los extremos para facilitar la transmisión de las cargas.
- Muelles de Torsión: Diseñados para funcionar frente a pares de torsión, es decir, un par de torsión se aplica a lo largo del eje del muelle, haciendo que el muelle se enrolle o se desenrolle. Estos muelles también tienen una variedad de formas y tamaños y a menudo tienen ganchos, lazos o extremos en espiral para facilitar su anclaje.
- Muelles Cónicos o de Forma Cónica: Son muelles helicoidales que tienen un diámetro que varía a lo largo de su longitud, lo que puede proporcionar una característica de carga no lineal cuando se comprimen o se estiran.
- Muelles de Placa o Hoja: Este tipo de muelle consiste en láminas de metal apiladas unas encima de otras, comúnmente usadas en la suspensión de los vehículos (Ballestas). El grosor, el ancho, y el número de láminas pueden variar, lo que permite diseñar una gran variedad de comportamientos de carga.
- Muelles de Anillo o de Forma de Arandela: Son muelles con forma de anillo, los cuales pueden comprimirse axialmente, proporcionando una fuerza de reacción.
- Muelles de Disco o Belleville: Son un tipo de muelle de arandela que proporciona una alta capacidad de carga en un espacio pequeño. Cuando se aplica una carga, el disco se aplana o se aplasta, proporcionando una fuerza de reacción.

Cada tipo de muelle tiene sus propias ventajas y desventajas, y la elección del tipo de muelle a utilizar dependerá del diseño específico y de las necesidades de la aplicación. El trabajo realizado se centra en analizar el comportamiento de los muelles helicoidales de compresión, pero la metodología sería aplicable al resto de configuraciones.

2.3.La constante de elasticidad

También conocida como constante de resorte o coeficiente de rigidez, es una propiedad física que describe la cantidad de deformación que un material experimentará bajo la influencia de una fuerza externa. Específicamente, en el contexto de los muelles, esta constante define cuánto se deformará un muelle cuando se le aplica una fuerza.

Esta constante se suele denotar con la letra 'k' y se define matemáticamente por la Ley de Hooke, que establece que la fuerza (F)

necesaria para comprimir o extender un muelle es proporcional a la distancia (x) que el muelle se ha comprimido o extendido desde su posición de reposo. Esto se puede expresar en la ecuación:

$$F = k \cdot x$$

Donde:

- F es la fuerza aplicada.
- k es la constante de elasticidad.
- x es la distancia que el muelle se ha comprimido o extendido.

La constante de elasticidad se mide en unidades de fuerza por longitud. En el sistema internacional (SI), estas unidades serían newtons por metro (N/m).

Un muelle con una alta constante de elasticidad requerirá más fuerza para ser comprimido o estirado una cierta distancia, en comparación con un muelle con una constante de elasticidad más baja. En otras palabras, un muelle con una constante de elasticidad alta es un muelle "más rígido" y un muelle con una constante de elasticidad baja es un muelle "más suave".

2.4.Investigaciones previas

En la literatura científica existen numerosos estudios que han explorado el análisis de muelles utilizando el método de los elementos finitos (FEM). Por ejemplo, el trabajo "FEM for Springs" ofrece un amplio análisis de la aplicación del FEM en la simulación y el diseño de diferentes tipos de muelles, destacando la flexibilidad y la precisión de este enfoque al abordar complejidades geométricas y diferentes tipos de cargas.

Además, un estudio específico para muelles de suspensión se presenta en "Finite Element Analysis of Helical Coil Compression Spring for Three Wheeler Automotive Front Suspension". En este trabajo, los investigadores realizaron un análisis FEM de un muelle helicoidal de compresión utilizado en la suspensión delantera de un vehículo de tres ruedas. Este estudio aporta un valioso ejemplo de cómo la metodología FEM puede usarse para simular el comportamiento y optimizar el diseño de un elemento mecánico, en este caso muelles de suspensión en la industria automotriz, resaltando tanto la versatilidad como la aplicabilidad práctica del FEM.

Estas publicaciones e investigaciones previas han servido para valorar las opciones a la hora de afrontar la creación de una herramienta que permita estimar la constante de elasticidad de un muelle genérico. Por ejemplo, en “FEM for Springs” muestran la metodología para simular el comportamiento de diversos diseños de resortes utilizando elementos 1D, 2D o 3D.

3.Metodología

3.1.Procedimiento de simulación

El problema que se resuelve en este trabajo, tiene varias facetas. El objetivo final del mismo es obtener una herramienta que nos obtenga la constante elástica de un muelle dadas sus propiedades geométricas y mecánicas, para realizar la resolución de este problema se ha dividido el trabajo en varias partes.

Como se ha descrito en capítulos anteriores, existen diversos tipos de muelles debido a su forma constructiva según la aplicación para la que son diseñados. En este trabajo, nos centramos en crear un método para obtener la constante elástica de un muelle helicoidal de sección circular.

El primer problema que se ha tenido que resolver, es la creación de la geometría. Por motivos que se discuten en capítulos posteriores, se ha planteado la resolución del problema por dos métodos diferentes, uno utilizando un modelo en 3D y otro método utilizando un modelo con elementos 1D.

Una vez generada la geometría del modelo a ensayar, esta se ha tenido que descomponer en elementos finitos. Para realizar estas dos tareas, se ha utilizado el software de modelado y mallado GMSH. Los diferentes pasos para generar la geometría se han resuelto con el programa creado utilizando la API de GMSH en lenguaje python.

Obtenida la malla con el modelo discretizado se ha procedido a configurar el análisis, resolver el problema planteado y obtener la solución a partir de los datos obtenidos. Como se muestra mas adelante, este análisis se realiza mediante el software Code Aster, un software que permite resolver mediante el método de los elementos finitos diferentes tipos de análisis, principalmente de índole mecánica y térmica.

Por último, la generación de la geometría, la malla, el lanzamiento del análisis y la obtención de resultados se ha automatizado mediante un programa generado en Python.

3.2.El software utilizado.

3.2.1.Gmsh

Gmsh es un software de generación de mallas 3D y preprocesamiento de elementos finitos de código abierto. Ofrece una serie de herramientas y características para ayudar en el diseño, procesado y análisis de modelos geométricos. Entre sus principales funcionalidades cabe destacar:

- Generador de Mallas: La principal funcionalidad de Gmsh es su capacidad para generar mallas 1D, 2D y 3D. Esto incluye mallas estructuradas y no estructuradas, permitiendo a los usuarios un control detallado sobre el tamaño y la forma de los elementos de la malla.

- Módulo CAD: Gmsh tiene un módulo CAD integrado que te permite crear y editar geometrías directamente en la interfaz de Gmsh. Este módulo incluye soporte para OpenCASCADE (OCC), que es una biblioteca de código abierto para modelado geométrico. OCC permite a Gmsh trabajar con formatos de archivo de CAD estándar y realizar operaciones complejas de modelado geométrico.
- Interfaz de Usuario: Gmsh ofrece tanto una interfaz gráfica de usuario (GUI) como una interfaz de línea de comandos, que facilitan la manipulación de la geometría y la malla y proporcionan una amplia gama de opciones para visualizar y examinar tu modelo.
- Interoperabilidad: Gmsh tiene soporte para una variedad de formatos de archivos, por lo que es compatible con muchos otros programas de simulación y análisis de elementos finitos. Esto hace que Gmsh sea muy flexible y útil en una variedad de flujos de trabajo de análisis.
- Lenguaje de script y API: Gmsh utiliza su propio lenguaje de script para automatizar muchas de las tareas de generación y manipulación de mallas. Además, Gmsh proporciona un API que permite su uso directamente desde otros lenguajes de programación, como Python, Fortran, Julia y C++. Esto permite a los usuarios automatizar y personalizar aún más su trabajo con Gmsh.
- Post-procesamiento: Aunque su principal fuerza es el preprocesamiento, Gmsh también tiene capacidades de post-procesamiento. Permite visualizar los resultados de las simulaciones, así como generar gráficos y animaciones.

En resumen, Gmsh es una herramienta de software potente y altamente personalizable que es útil para una amplia variedad de aplicaciones en análisis de elementos finitos y otras áreas de ingeniería y física.

3.2.2.Code_Aster

Code_Aster es un software de simulación que utiliza el método de elementos finitos de código abierto que se utiliza para realizar análisis estructurales y térmicos. El software fue desarrollado inicialmente por Électricité de France (EDF), la principal empresa de energía de Francia, y ha estado en desarrollo continuo durante más de 30 años.

Las principales características de Code_Aster incluyen:

- Análisis de estructuras y materiales: Code_Aster es capaz de realizar análisis estáticos lineales y no lineales, análisis dinámicos, y también tiene capacidades para el análisis de daños y fallos en estructuras.
- Diversidad de elementos finitos: Soporta una amplia gama de tipos de elementos finitos, incluyendo elementos 1D (como barras y vigas), 2D (como placas y carcasas), y 3D (como hexaedros y tetraedros).
- Resolución de problemas multiphysiques: Code_Aster también es capaz de resolver problemas multiphysiques, lo que permite el análisis de

fenómenos acoplados, como termo-mecánica, acústica, o flujo de fluidos y transferencia de calor.

- Manipulación de datos: El software proporciona funcionalidades robustas para la importación y manipulación de datos, y para la visualización y el postprocesamiento de resultados.
- Compatibilidad con Salome: Code_Aster está diseñado para funcionar bien con Salome, que es un software de código abierto para el pre y post procesado en análisis. Juntos, estos dos programas forman una solución completa para el modelado, simulación y análisis de elementos finitos.
- Código abierto: Como software de código abierto, Code_Aster permite a los usuarios adaptar y personalizar el software según sus necesidades. También se beneficia de una comunidad activa de usuarios que contribuyen al desarrollo y mejoramiento del software.

En resumen, Code_Aster es un software de simulación de elementos finitos muy versátil y poderoso que puede ser utilizado para un amplio rango de aplicaciones en ingeniería y física.

3.3.La geometría.

En un muelle helicoidal de compresión, se pueden distinguir dos partes. Por un lado, el cuerpo del muelle, que es la parte encargada de deformarse cuando se ejerce fuerza de compresión entre su dos extremos. En los muelles de compresión helicoidales esta deformación se produce en la dirección longitudinal del mismo. En el mercado, se encuentran diversos tipos de muelles según la geometría de su sección, pero este trabajo se centra en resolver el problema en muelles de sección circular maciza.

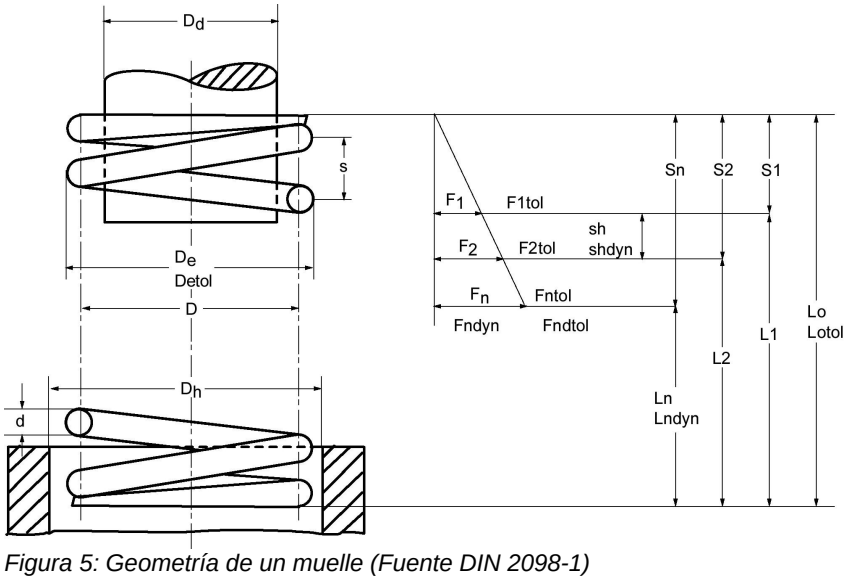

Figura 5: Geometría de un muelle (Fuente DIN 2098-1)

Por otro lado, tenemos los extremos. A la hora de colocar el muelle, este tiene que ir guiado por su interior o por su exterior para evitar cargas en sentido radial, y así las fuerzas ejercidas sobre el mismo serán estrictamente de

compresión. Estos extremos pueden presentar diversas formas constructivas dependiendo de la forma de sus extremos.

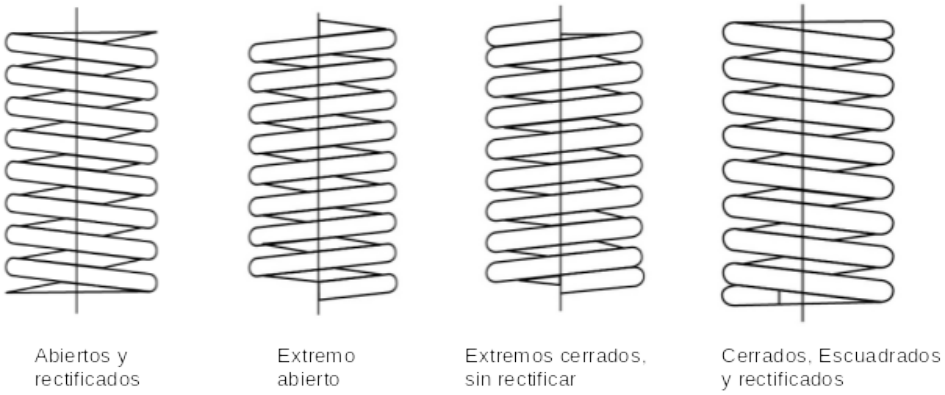

Figura 6: Tipos de extremos muelles helicoidales (Fuente www.leespring.es [11])

Se ha de tener en cuenta que la forma del extremo no tiene una influencia relevante sobre los resultados obtenidos, ya que la deformación de las espiras de los extremos es varios órdenes de magnitud inferior a la deformación de las espiras que conforman el cuerpo del muelle. En la realización de este estudio, se ha realizado el modelo con ambos extremos cerrados, sin rectificar.

A la hora de realizar el modelo lo primero es crear los puntos sobre los que se apoya la entidad geométrica sobre la que se construye el muelle helicoidal: el helicoide. Estos puntos se crean dividiendo la longitud del alambre del muelle en segmentos de igual longitud distribuidos uniformemente a lo largo de las espiras activas y las espiras de los extremos.

El segundo paso que realiza la rutina es crear una curva de bezier (spline) a la cual pertenecen todos los puntos generados. En el caso de la rutina 1D, aquí terminaría el modelo geométrico.

Para poder realizar el análisis con elementos 3D se tiene que generar un cuerpo sólido. Para ello, se crea una superficie en forma de disco en la mitad del segmento con una orientación perpendicular a la espira. La rutina utiliza la función *gmsh.model.occ.addPipe* para generar una extrusión de la superficie de la sección a lo largo de las hélice del cuerpo y los extremos.

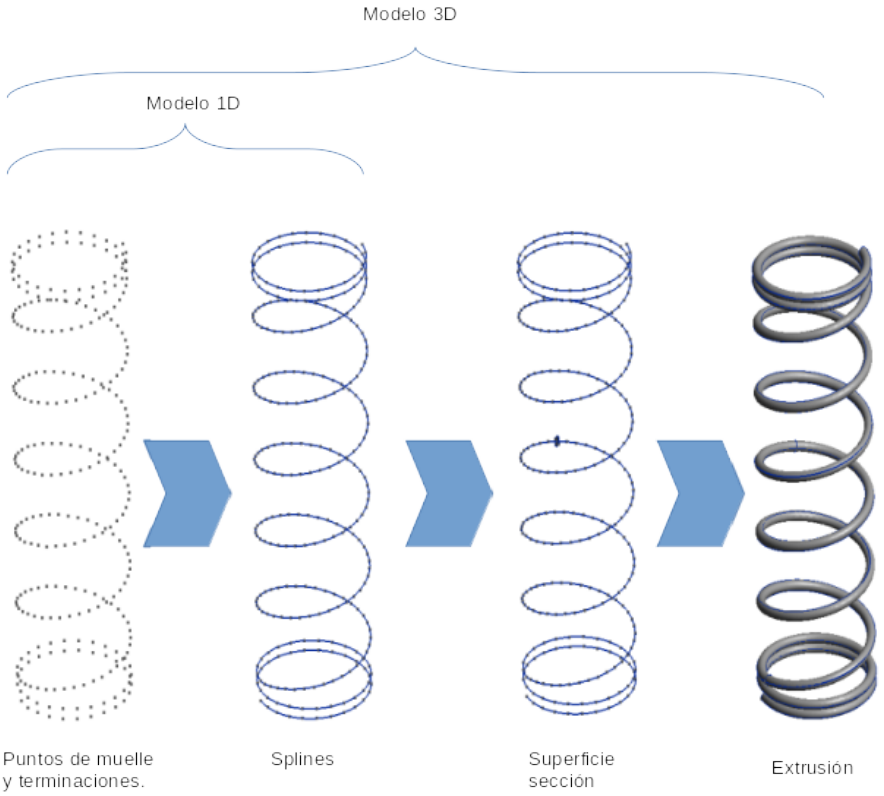

Figura 7: Proceso de creación de la geometría

3.4.El mallado.

Una vez concluida la parte de generación de los modelos, la rutina crea la malla. Uno de los principales factores para hacer un análisis y que los resultados sean coherentes es que la malla tenga la dimensión correcta. Como se verá en capítulos posteriores, el tamaño de la malla está directamente relacionado con la calidad de los resultados. La limitación del tamaño de malla varía en función del modelo que se quiera usar.

Si el problema lo resolvemos por elementos unidimensionales, la longitud de cada elemento viene limitado por el ratio del diámetro del hilo y el factor de forma que configuremos. En este modelo se ha utilizado elementos viga, y su tamaño mínimo permitido es de ⅓ del radio del alambre de la espira. Con tamaños de malla mas pequeños, el solver devuelve un error por factor de forma no permitido y como veremos mas adelante, los resultados ofrecen poca variación si reducimos el tamaño de los elementos.

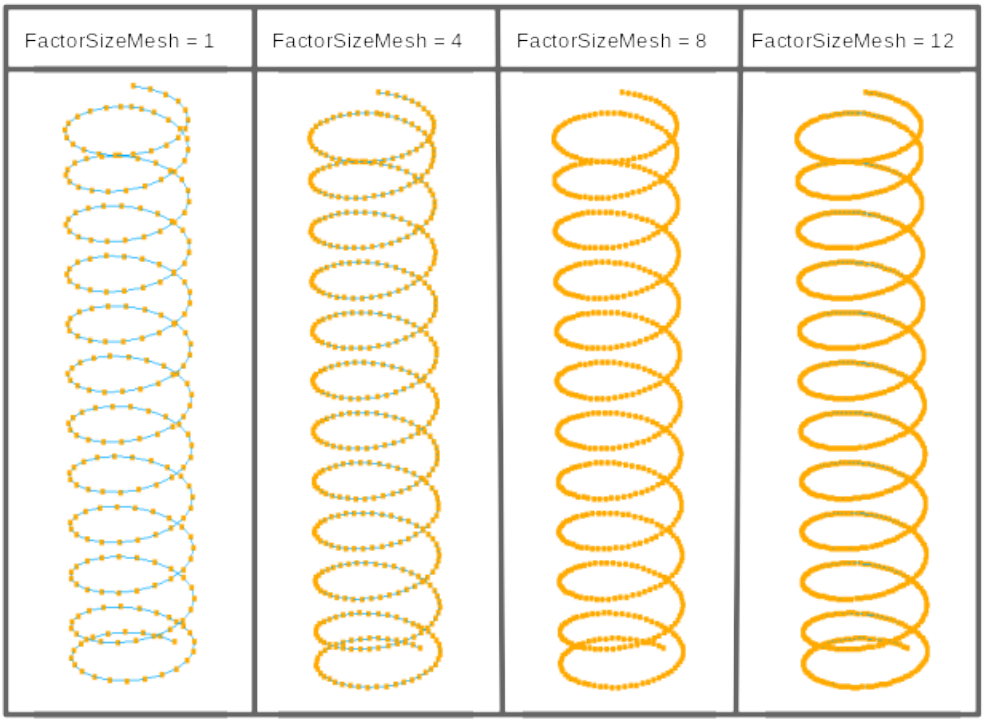

Figura 8: Diferentes tamaños de malla caso 1D

Para resolver el problema mediante elementos de tres dimensiones se ha tomado la misma determinación. Se ha vinculado el tamaño de malla con el diámetro del alambre.

En ambos casos esta relación entre ambos parámetros se regula con la variable “FactorMeshSize” que como se muestra con posterioridad, el tamaño de malla influye notablemente en la calidad de los resultados, a costa de aumentar los tiempos e cálculo.

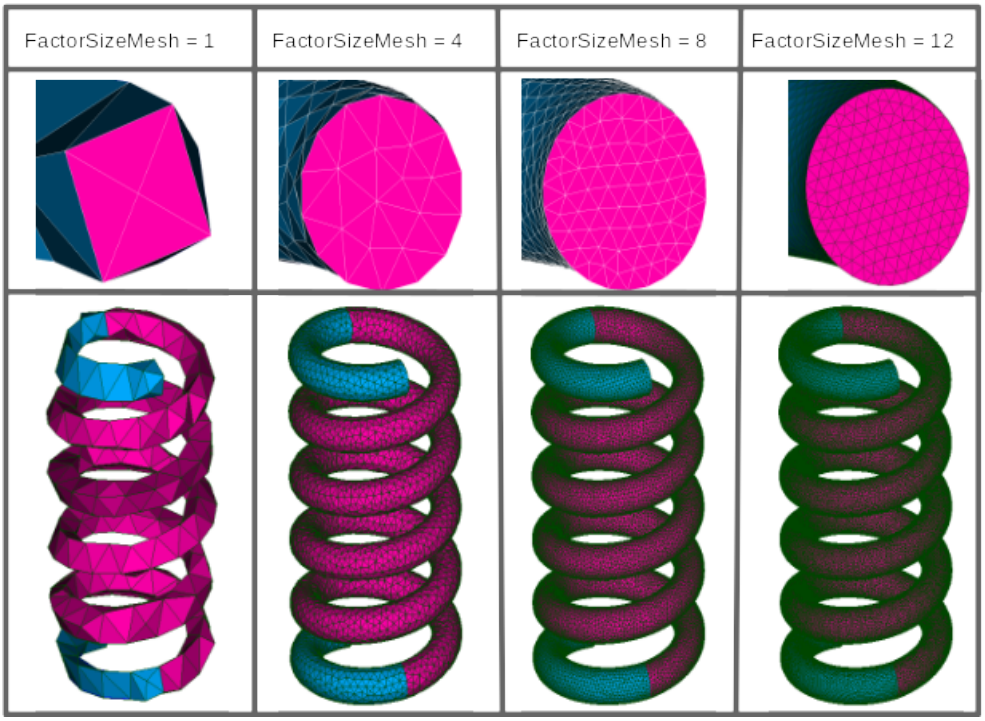

Figura 9: Diferentes tamaños de malla caso 3D en función de el parámetro "FactorMeshSize".

Las mallas de ambos modelos están formadas por los siguientes elementos:

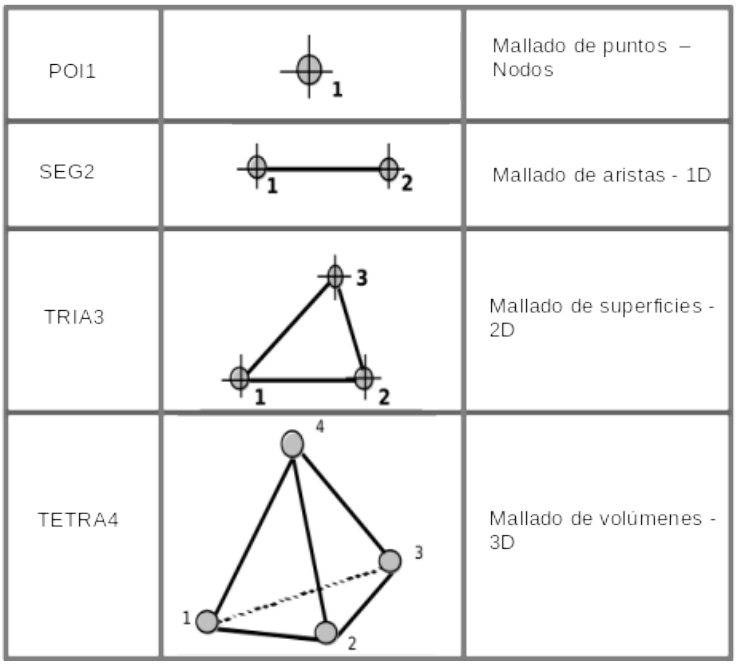

Figura 10: Elementos utilizados en los modelos.

Estos elementos son creados en orden para cada uno de los casos. En el caso del análisis 1D, primero se crean los nodos sobre la spline. El siguiente paso que ejecuta el software es realizar el mallado en la spline con elementos 1D.

En el caso del modelo 3D, sobre las aristas del volumen que forma el muelle GMSH genera primero los nodos. Una vez generados los nodos, crea los elementos barra. Una vez que tiene los elementos barra, realiza el mallado de la superficies, para acabar con la discretización del volumen del sólido.

En este caso no se encuentra una limitación de tamaño debido a ningún factor de forma del elemento. Como se verá mas adelante, el criterio para decidir el tamaño de malla viene dado por los recursos de cálculo que se desee utilizar.

3.5.Descripción de los modelos de simulación utilizados.

Para la estimación de la constante elástica del muelle, se utilizó el modelo MECA_STATIQUE proporcionado por Code_Aster [9]. Este modelo resuelve las ecuaciones de equilibrio estático que surgen del principio de los trabajos virtuales en su forma discreta, expresadas en su forma matricial:

$$[K]\{u\}=\{F\}$$

En esta ecuación, [K] es la matriz de rigidez global, construida a partir de las matrices de rigidez de los elementos individuales y que depende de las propiedades elásticas del material y de la geometría del muelle; {u} es el vector de desplazamientos nodales desconocidos que se obtiene como resultado de la solución del sistema; y {F} es el vector de fuerzas nodales conocidas, representado en este caso por una carga nodal de 1N aplicada en un extremo del muelle en dirección Z.

La resolución de este sistema de ecuaciones se realiza mediante un algoritmo numérico nos permite resolver un problema mecánico con carga estática y diversas condiciones de contorno y aplicación de cargas.

En cuanto a las condiciones de contorno, se han restringido los desplazamientos en X e Y en el extremo donde se aplica la carga y se han fijado los desplazamientos en todas las direcciones en el extremo opuesto. Estas restricciones son necesarias para simular correctamente las condiciones reales de funcionamiento de un muelle evitando giros en las zonas de aplicación de cargas y restricciones.

Se consideraron dos enfoques de modelado: unidimensional (1D) y tridimensional (3D). El modelado 1D, aunque más simple y computacionalmente menos costoso, puede no capturar todos los detalles del comportamiento real del muelle. Por otro lado, el modelado 3D, a pesar de su mayor costo computacional, permite una representación más precisa del muelle y puede proporcionar resultados más acertados.

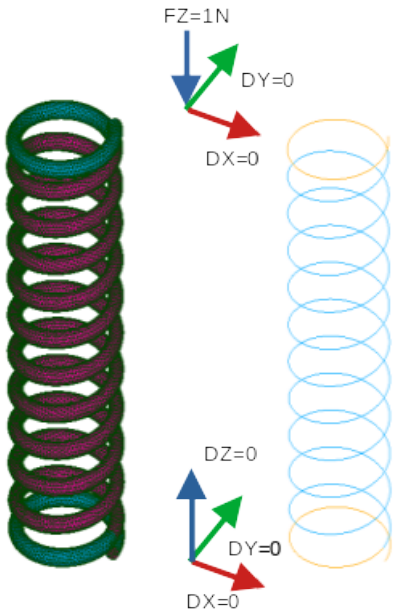

Figura 11: Fuerzas y restricciones aplicadas al modelo.

Es importante mencionar que el tamaño de la malla utilizada en las simulaciones tiene un impacto significativo tanto en la precisión de los resultados como en los tiempos de cálculo. Generalmente, una malla más fina da lugar a resultados más precisos, pero también requiere un tiempo de cálculo más largo. Por tanto, se debe buscar un equilibrio entre precisión y eficiencia computacional.

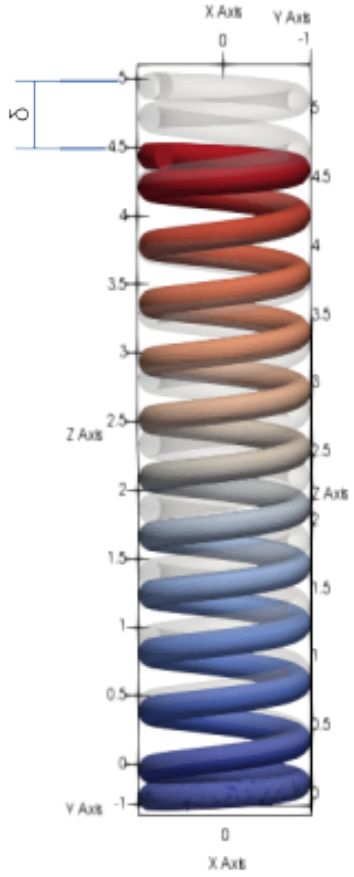

Figura 12: Obtención del máximo desplazamiento

Finalmente, la constante elástica del muelle se puede calcular a partir del desplazamiento en la dirección Z obtenido en el nodo donde se produce mayor desplazamiento. Utilizando la ley de Hooke (F = kx), donde F es la fuerza aplicada y x es el desplazamiento, se puede obtener de forma sencilla el valor de k.

3.6.Selección de parámetros y variables de entrada

En este capítulo, se analiza en detalle la selección de los parámetros y variables de entrada utilizados en las funciones "lanzador3D" y "lanzador1D" que fueron desarrolladas para calcular la constante elástica del muelle. Estas funciones se define como:

- "def lanzador3D(nturns, dmuelle, dalambre, hmuelle, nmuelle, bturns, FactorMeshSize, E, NU):"
- "def lanzador1D(nturns, dmuelle, dalambre, hmuelle, nmuelle, bturns, FactorMeshSize, E, NU):"

Existen tres tipos principales de variables de entrada que se utilizan en estas funciones:

Datos geométricos: Estos parámetros describen la geometría del muelle y son fundamentales para su modelado preciso. Los parámetros incluyen:

- nturns:El número de espiras en el muelle.
- dmuelle: El diámetro exterior del muelle.
- dalambre: El diámetro del alambre utilizado para el muelle.
- hmuelle: La altura del muelle entre las espiras activas.
- bturns: El número de espiras en la base del muelle.

Propiedades del material: Estos parámetros definen las características elásticas del material utilizado en el muelle. Los parámetros son:

- E: El límite elástico del material. Representa la capacidad del material para resistir deformaciones sin sufrir daños permanentes.
- NU: El coeficiente de Poisson. Indica la relación entre las deformaciones longitudinales y transversales del material.

Variables adicionales:

- nmuelle: Es el nombre asignado al muelle, utilizado para su identificación dentro del programa.
- FactorMeshSize: Es un factor de tamaño de malla utilizado en el proceso de mallado del modelo. Este parámetro permite ajustar la densidad de la malla, lo cual influye en la precisión de los resultados obtenidos y en los tiempos de cálculo. Al realizar el ajuste del parámetro "FactorMeshSize" se debe considerar la necesidad de lograr un equilibrio entre la precisión requerida y la eficiencia computacional, ya que un tamaño de malla más fino ofrece resultados más precisos, pero aumenta el tiempo de cálculo.

En los modelos de 1D y 3D el parámetro FactorMeshSize permite los siguientes valores de malla máximos y mínimos:

- MeshSizeMin = dalambre*1/(FactorMeshSize*3)
- MeshSizeMax = dalambre*1/FactorMeshSize

Hay que tener en cuenta que el valor de MeshSizeMax está limitado a un sexto del díametro de la espira, ya que como indica en la documentación de code_aster, si el tamaño del elemento es menor que la tercera parte del radio característico de la barra aparece un error de forma en el modelo y no puede realizar el análisis.

3.7.Validación del modelo

En este capítulo, se aborda la validación del modelo mediante la comparación de los resultados obtenidos con resultados teóricos. Esta validación permite evaluar la precisión y confiabilidad de los resultados obtenidos a través de las funciones "lanzador3D" y "lanzador1D" en el cálculo de la constante elástica del muelle.

La validación se realiza comparando los resultados obtenidos mediante la función "lanzador3D" mediante dos métodos:

- El primero, con los valores de la constante elástica proporcionados por los fabricantes de los muelles comerciales
- El segundo, con los resultados obtenidos mediante la ley de la deformación, mediante la cual se puede calcular la deformación que experimenta el muelle helicoidal en función de la carga axial aplicada en sus extremos.

Tal como es descrito en Gamm [2] y Shimoseki [1], del desarrollo del teorema de la ley de la deformación se desprende que la constante de un muelle helicoidal trabajando a compresión se aproxima como función de la geometría y el material del mismo. Este método estima la constante elástica de un muelle mediante la estimación de la deformación.

La necesidad de utilizar este método en conjunto con el valor dado por el fabricante de los muelles ensayados (ambos valores coinciden, ya que los fabricantes estiman el valor de K mediante este método) es para tener una referencia de valor de la constante elástica cuando no se tiene una estimada por el fabricante. De esta forma podemos validar resultados en muelles personalizados para aplicaciones específicas, con propiedades y geometrías no existentes en los catálogos de los fabricantes.

$$k = \frac{Gd^4}{8nD^3},$$

donde:

- k = constante elástica
- d = diámetro del alambre
- D = diámetro de la espira
- n = número de espiras activas
- G = módulo de cortadura

Hay que tener en cuenta que este resultado nos dará un valor de constante elástica que también es una aproximación, ya que en la obtención de esta expresión se han tenido una serie de suposiciones:

Las suposiciones utilizadas en la derivación de la ecuación son:

- No se considera el efecto de una viga curvada ubicada especialmente. El modelo se construye como la torsión de una barra recta.
- La fuerza externa se aplica a lo largo del eje central de la bobina. Se desprecian las cargas o momentos oblicuos excéntricos desde los extremos.
- Se asume que la deformación es pequeña. La deflexión se considera insignificante en comparación con la dimensión del resorte.

La validación del modelo es un paso esencial para asegurar la precisión y confianza de los resultados obtenidos. Permite evaluar si el modelo implementado en el programa es capaz de capturar de manera adecuada el comportamiento elástico del muelle y proporcionar resultados coherentes.

Si los resultados obtenidos mediante las funciones “lanzador1D” y "lanzador3D" están en concordancia con los resultados obtenidos mediante la ley de deformación, se refuerza la validez y precisión del modelo implementado.

Es importante destacar que la validación del modelo debe realizarse utilizando varios casos de estudio que abarquen diferentes geometrías y propiedades de materiales. Esto permitirá evaluar la capacidad del modelo para adaptarse a diferentes situaciones y garantizar su aplicabilidad en un amplio rango de escenarios.

En resumen, la validación del modelo se lleva a cabo comparando los resultados obtenidos mediante la funciones con los valores de la constante elástica proporcionados por los fabricantes de muelles comerciales y con los resultados obtenidos mediante la ley de deformación. Esta comparación permite evaluar la precisión y confianza del modelo implementado en el cálculo de la constante elástica del muelle.

4.Resultados y análisis

En este apartado, se presentan los resultados obtenidos a partir de las simulaciones y análisis realizados utilizando diferentes modelos de muelles y tamaños de malla. Se realizaron comprobaciones con cuatro modelos de muelles diferentes, dos de cada fabricante, con el objetivo de comparar los resultados y evaluar la precisión del método utilizado para calcular la constante elástica del muelle. Además, se ha realizado la evaluación del tiempo de computación requerido para cada caso.

4.1.Casos de estudio

En este apartado se detallan los casos de estudio realizados. Se han llevado a cabo diversas simulaciones utilizando cuatro modelos de muelles diferentes. Dos de ellos parten del catálogo del fabricante de muelles LeeSpring, mientras que las propiedades de los otros dos son extraídos del catálogo proporcionado por el fabricante Lesjöfors.

Se ha de tener en cuenta que los datos geométricos utilizados deben ser valorados de forma crítica antes de ser utilizados ya que los fabricantes utilizan nomenclaturas parecidas para diferentes dimensiones en los parámetros de cada muelle. En este sentido, los parámetros utilizados en las funciones se han tenido que revisar y procesar para que sean coincidentes con los parámetros dados por los fabricantes.

A continuación, se presenta una tabla con las dimensiones y el material de cada muelle, así como una tabla adicional con las propiedades específicas de cada material utilizado.

Dimensiones y materiales de los muelles:

nmuelle	nturns	bturns	dmuelle	dalambre	hmuelle	Material
M6925	4.5	1	79	14	120	EN 10270-1-SH
M1117	10.1	1	1.52	0.2	4.8	EN 10270-1-SH
CID040EG	5.5	1	5.4	0.4	15.2	ASTM A313
LHC_207U_03S	5.15	1	49.2	5.26	73.12	ASTM A313

Y las propiedades de los materiales utilizados son:

Material	E	ν
EN 10270-1-SH	206000	0.2644
ASTM A313	190482,46	0,4

Para cada uno de estos modelos de muelles, se han lanzado diversos casos de estudio para cada uno, variando el parámetro “FactorMeshSize” para realizar el análisis con diferentes valor de parámetro y poder valorar su precisión y el coste de cálculo de una malla de elementos de diferentes tamaños. El parámetro se introduce como valor entero y se han realizado análisis variando su valor desde 1 hasta valor 20.

Además, para disminuir la incertidumbre producida en cada uno de los cálculos, se han realizado 5 cálculos para cada caso planteado, obteniendo una media aritmética de cada valor (Constantes elásticas y tiempos invertidos) que será utilizada para su análisis.

4.2.Análisis de los resultados obtenidos

4.2.1.Valores obtenidos de constante elástica

En este apartado, se presentan las gráficas comparativas que muestran la convergencia de los valores de la constante elástica obtenidos mediante los modelos 1D (K1D) y 3D (K3D) hacia el valor teórico (KT) para cada muelle analizado. También se encuentran las gráficas comparando los tiempos de cálculo de ambos métodos. Son analizados en su conjunto, pero también se presentan separando los tiempos utilizados para la creación de la geometría y los tiempos utilizados para la realización del análisis.

Para cada muelle estudiado, se muestra una gráfica que compara los valores de KT, K1D y K3D en función del tamaño de malla (FactorMeshSize). Estas gráficas permiten visualizar la convergencia de los resultados hacia el valor teórico y analizar la discrepancia entre los valores obtenidos mediante los distintos modelos.

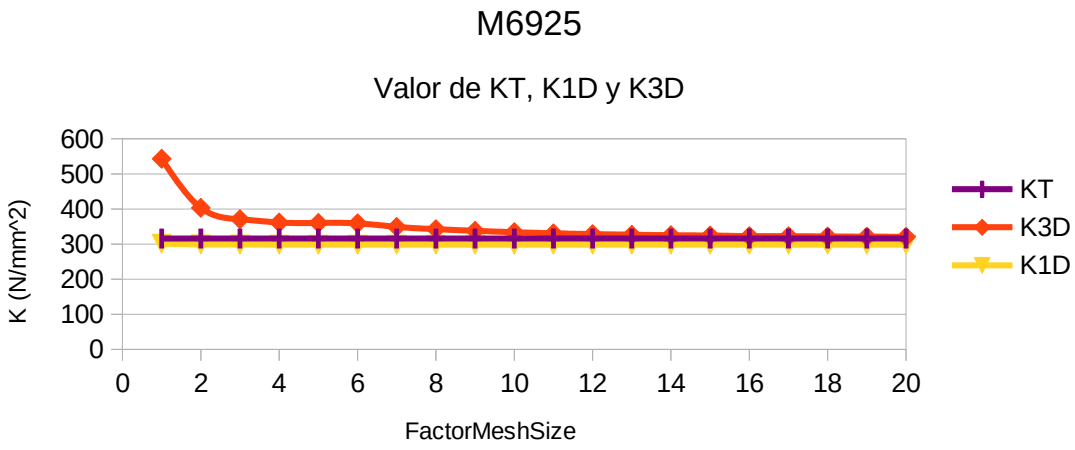

Figura 13: Gráfica comparativa M6925 KT, K1D y K3D diferentes valores de malla.

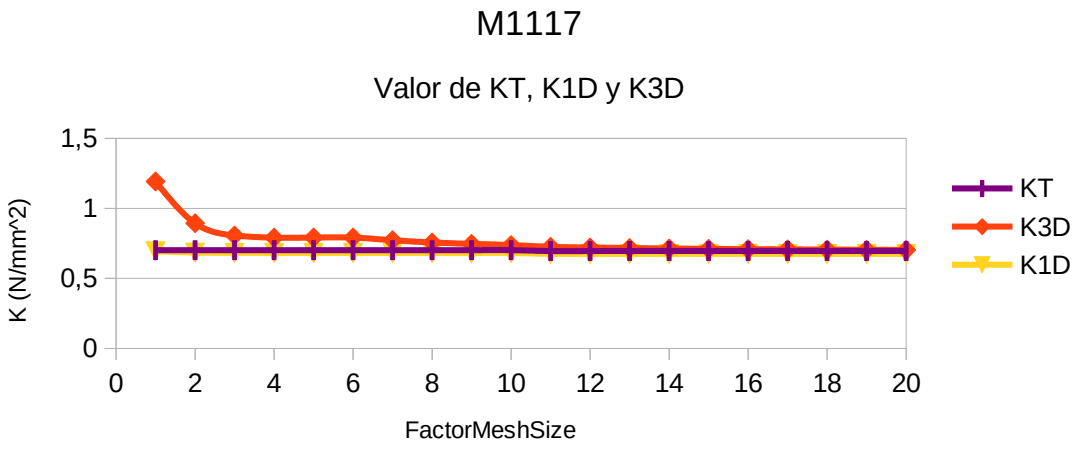

Figura 14: Gráfica comparativa M1117 KT, K1D y K3D diferentes valores de malla.

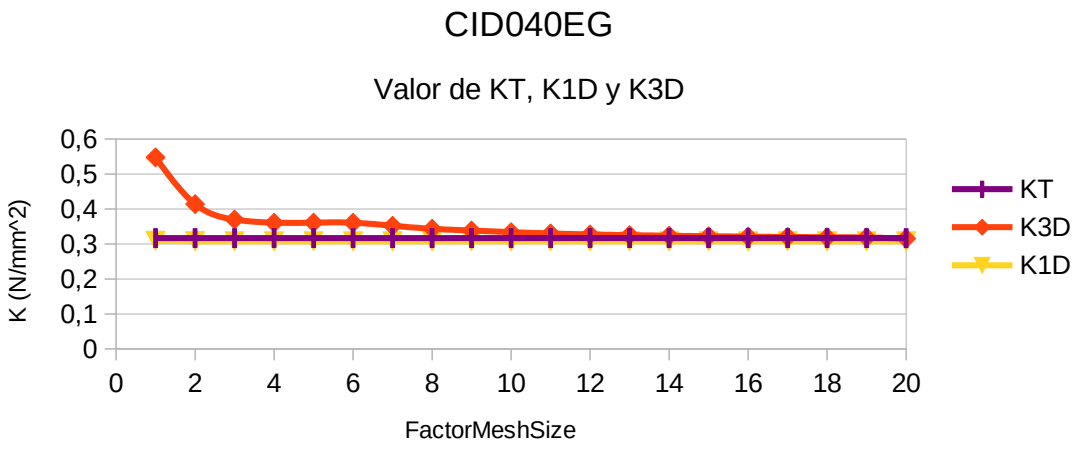

Figura 15: Gráfica comparativa CID040EG KT, K1D y K3D diferentes valores de malla.

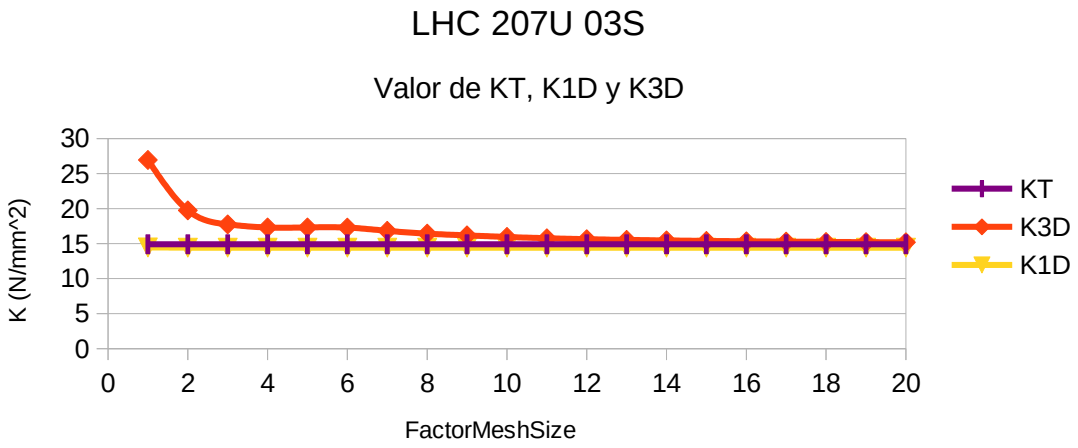

Figura 16: Gráfica comparativa LHC 207U 03S KT, K1D y K3D diferentes valores de malla.

Se observa que tanto K1D como K3D convergen hacia un valor cercano al valor teórico KT. En el caso de K3D, se acerca progresivamente a KT desde arriba y converge hacia él a medida que se reduce el tamaño de malla. Sin embargo, se observa un aumento exponencial en los tiempos de cálculo a medida que se incrementa el parámetro del tamaño de malla en K3D. Por otro lado, K1D converge hacia un valor cercano a KT desde abajo, pero los tiempos de cálculo son similares para cualquier valor de tamaño de malla usado.

4.2.2.Valores de los tiempos de creación de geometría y mallado en GMSH

En este apartado se muestran las gráficas del análisis realizado del estudio de los tiempos invertidos en la creación de la geometría y el mallado para cada uno de los modelos de muelle según se ha modificado el factor de forma.

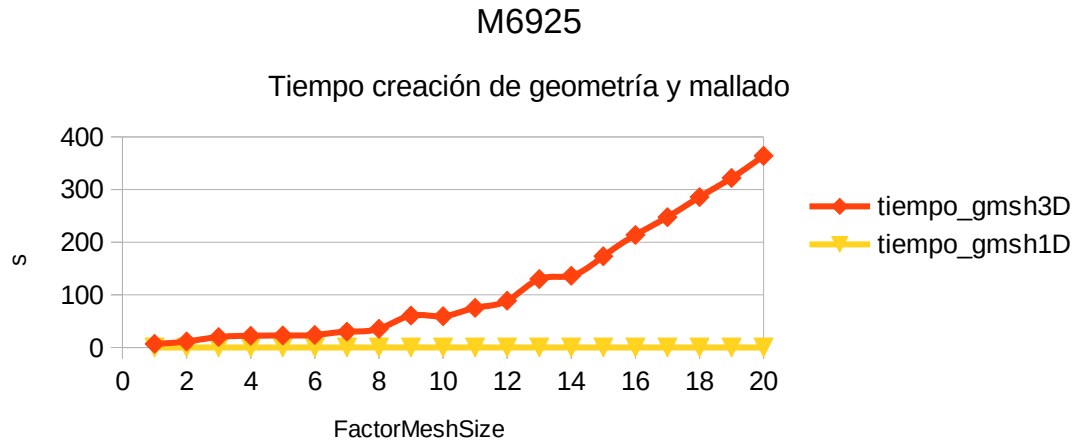

Figura 17: Gráfica comparativa M6925 tiempos de GMSH 1D y 3D

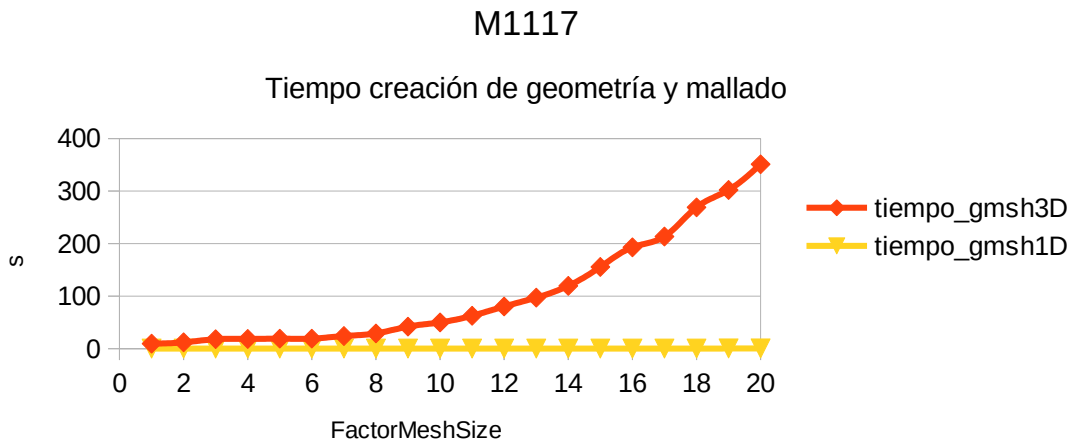

Figura 18: Gráfica comparativa M1117 tiempos de GMSH 1D y 3D

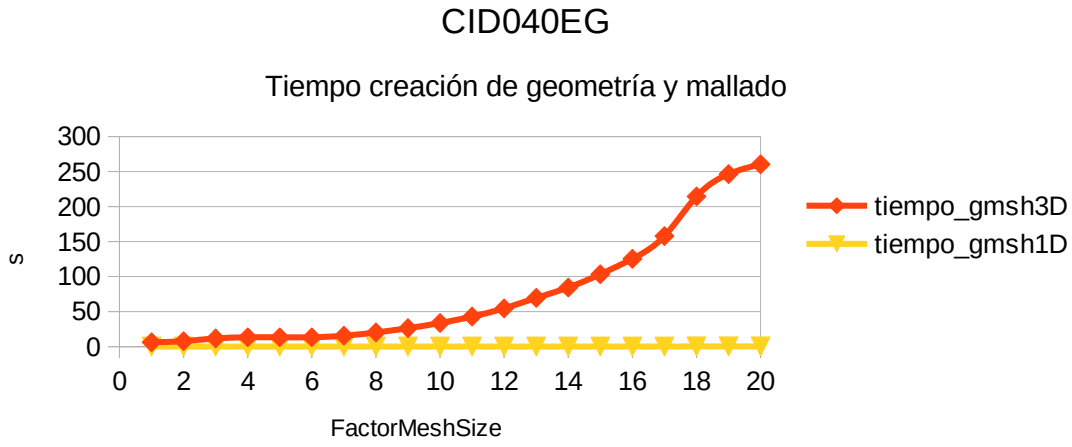

Figura 19: Gráfica comparativa CID040EG tiempos de GMSH 1D y 3D

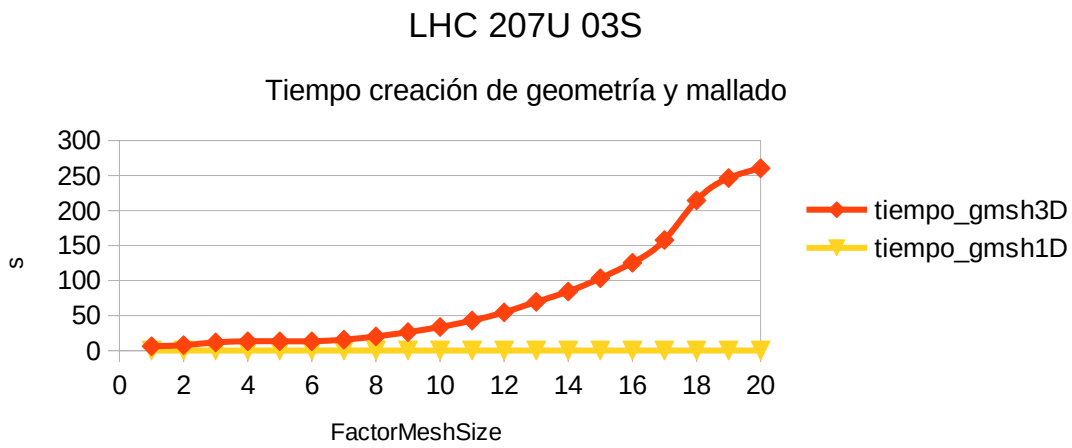

Figura 20: Gráfica comparativa LHC 207U 03S tiempos de GMSH 1D y 3D

Como se puede observar, los tiempos utilizados en el método 3D aumentan exponencialmente frente a los tiempos invertidos en el método 1D, que apenas varían. También podemos observar como en el modelado y mallado, los tiempos entre diferentes geometrías son similares.

4.2.3.Valores de los tiempos de cálculo en code_aster

En este apartado se muestran las gráficas del análisis realizado del estudio de los tiempos invertidos en la resolución del problema mecánico y l obtención de resultados para cada uno de los modelos de muelle según se ha modificado el factor de forma.

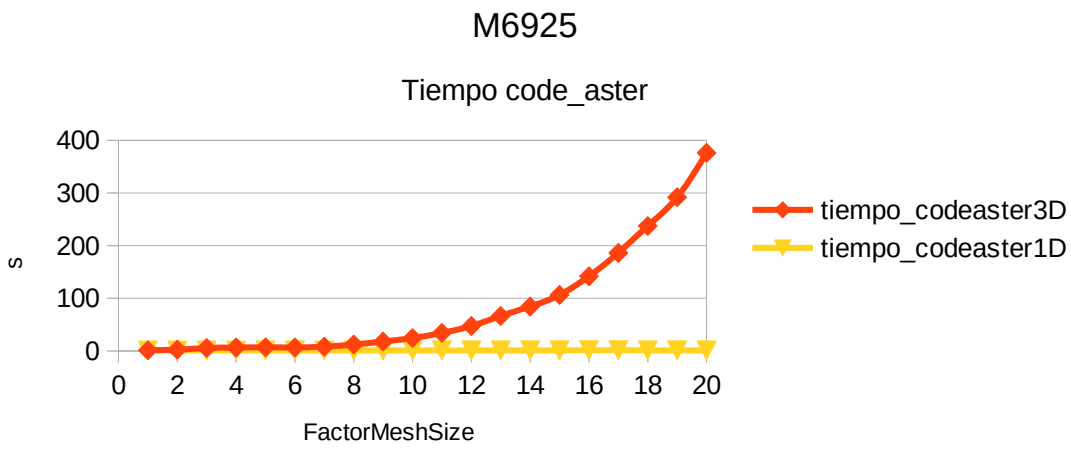

Figura 21: Gráfica comparativa M6925 tiempos de Code_Aster 1D y 3D

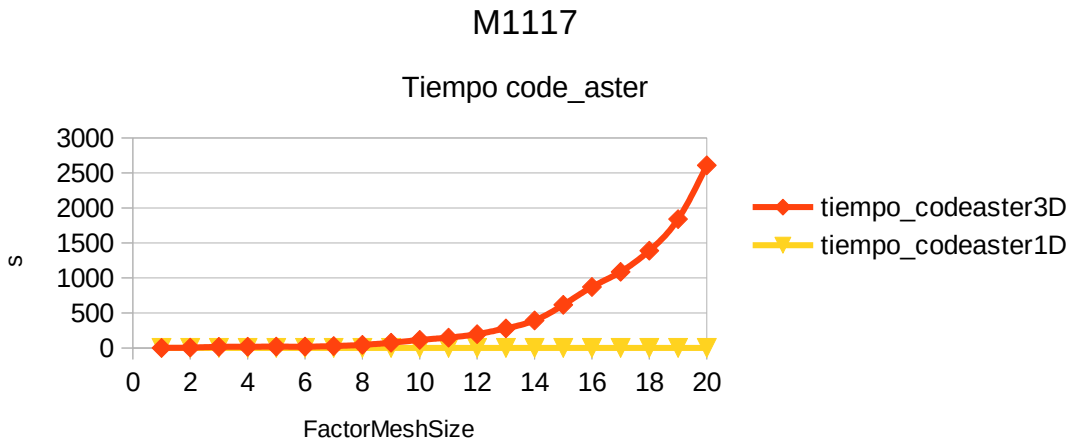

Figura 22: Gráfica comparativa M1117 tiempos de Code_Aster 1D y 3D

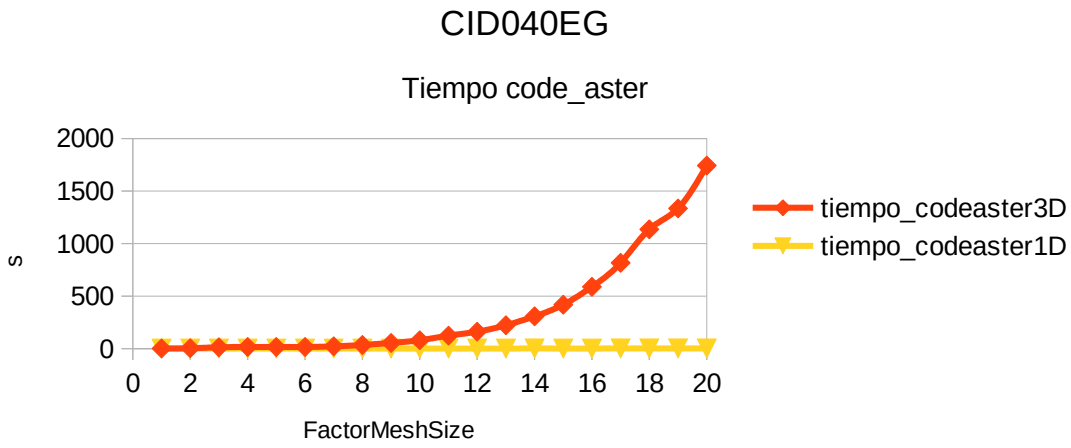

Figura 23: Gráfica comparativa CID040EG tiempos de Code_Aster 1D y 3D

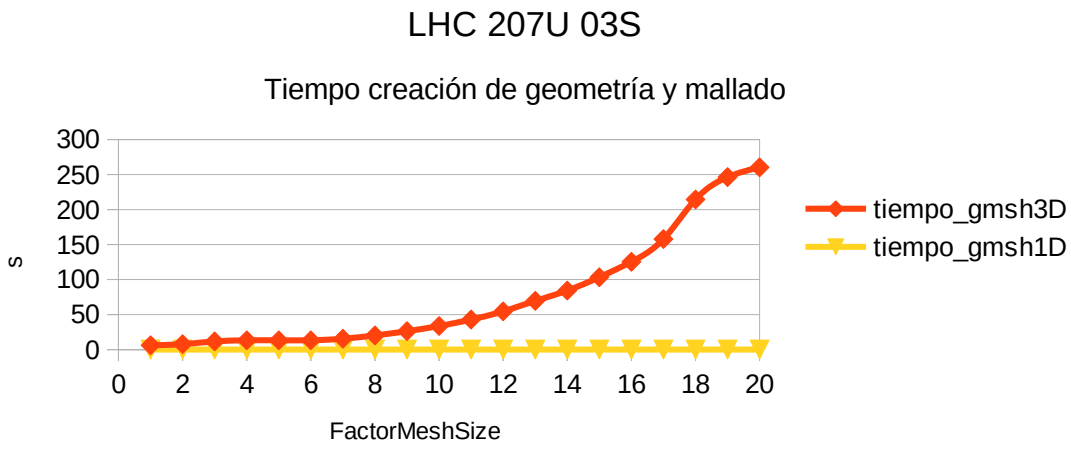

Figura 24:Gráfica comparativa LHC 207U 03S tiempos de Code_Aster 1D y 3D

De nuevo se puede observar como los tiempos utilizados en el método 3D aumentan exponencialmente frente a los tiempos invertidos en el método 1D, que apenas varían. En este apartado se puede observar como los tiempos de cálculo si varían de un modelo a otro de forma sustancial. Como se puede ver en capítulos anteriores, el tamaño de malla se relaciona con el diámetro de espira. Esto hace que en muelles semejantes los tiempos de cálculo sean similares, ya que el número de elementos está en el mismo orden de magnitud. Sin embargo, cuanto mayor es el número de elementos y nodos que componen el modelo, la matriz a resolver es mayor, aumentando así los tiempos de procesamiento y resolución del sistema.

4.2.4.Valores del tiempo total

En este apartado se muestran en análisis realizado del estudio de los tiempos de totales invertidos para cada uno de los modelos de muelle según se ha modificado el factor de forma.

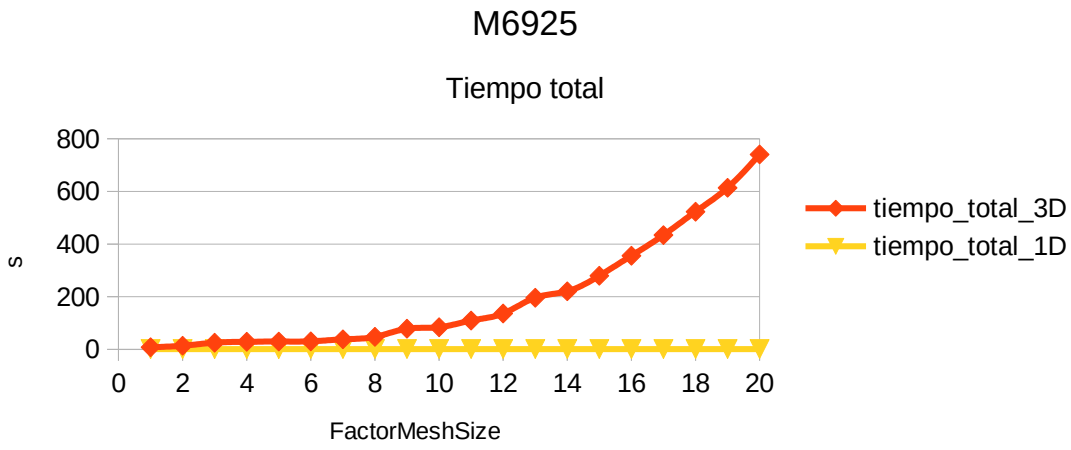

Figura 25: Gráfica comparativa M6925 tiempos totales 1D y 3D

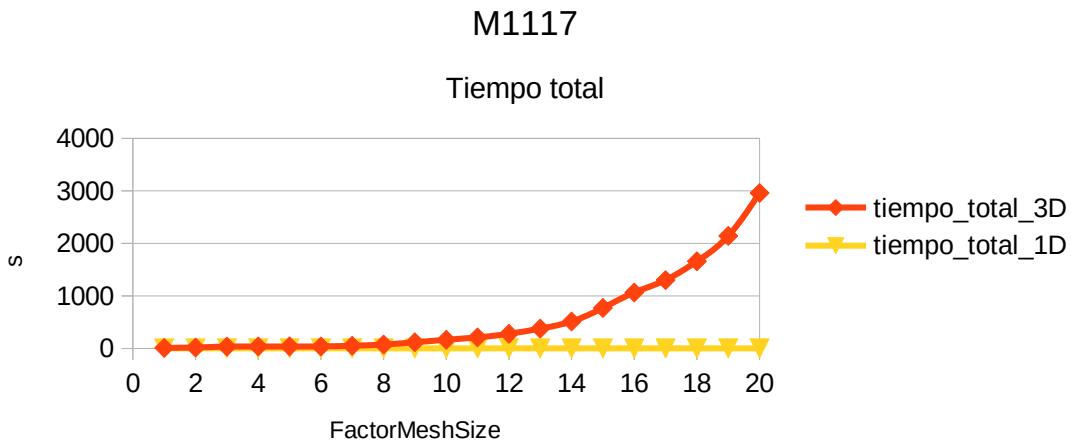

Figura 26: Gráfica comparativa M1117 tiempos totales 1D y 3D

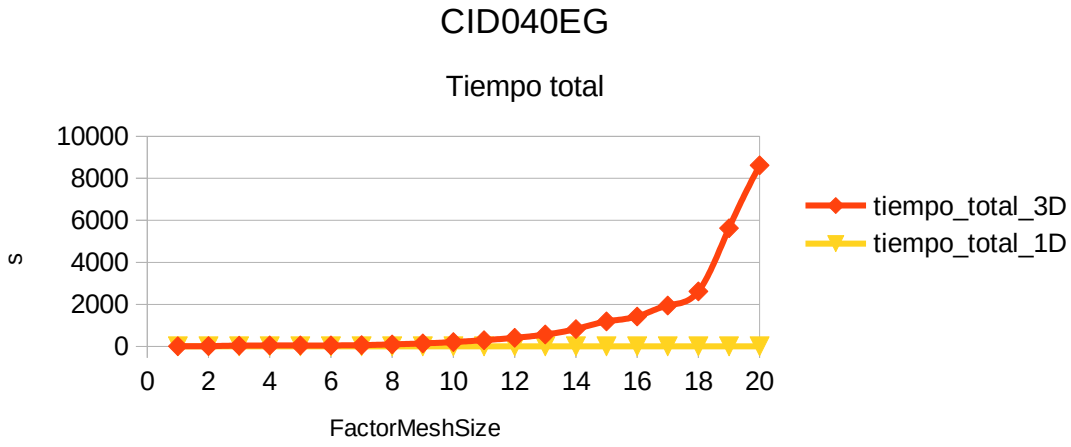

Figura 27:Gráfica comparativa CID040EG tiempos totales 1D y 3D

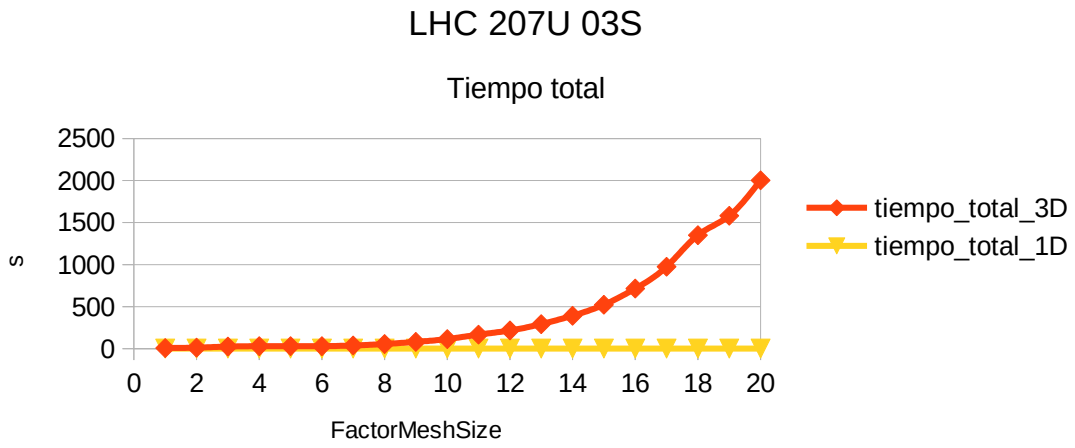

Figura 28:Gráfica comparativa LHC 207U 03S tiempos totales 1D y 3D

Finalmente, estos resultados nos dan la comparación de los tiempos totales invertidos en realizar cada análisis.

4.2.5.Análisis de la desviación en los resultados con KT

Por último, se ha realizado el análisis de los resultados obtenidos comparando ambos métodos con el valor de teórico de la constante, mostrado como desviación porcentual frente a KT.

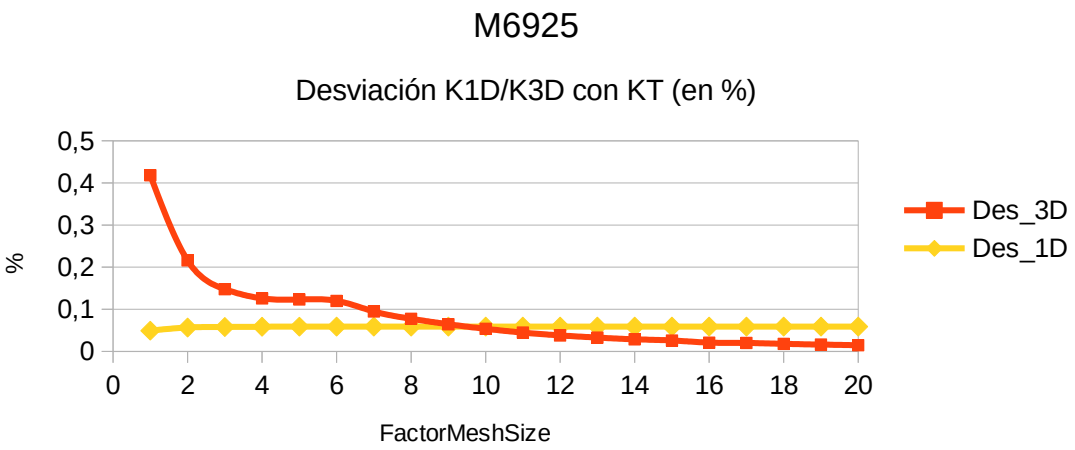

Figura 29:Gráfica comparativa M6925 Desviación K3D/K1D vs KT

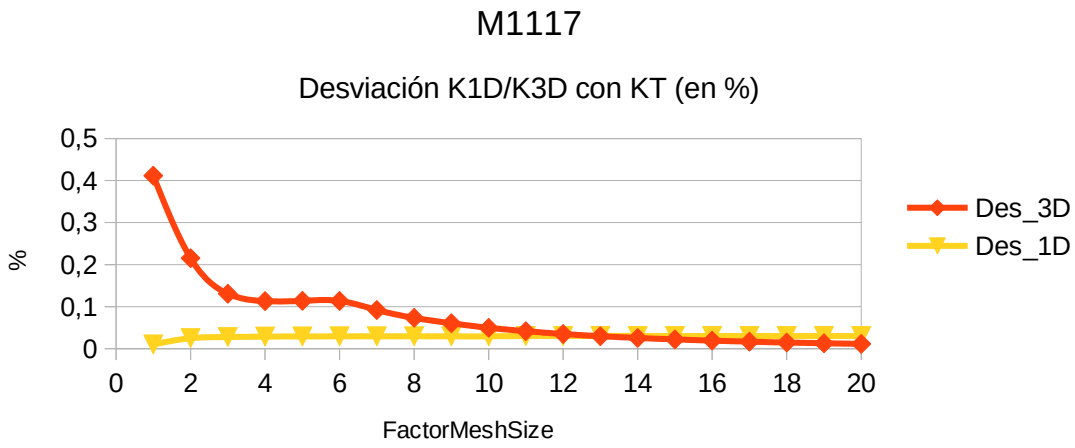

Figura 30: Gráfica comparativa M1117 Desviación K3D/K1D vs KT

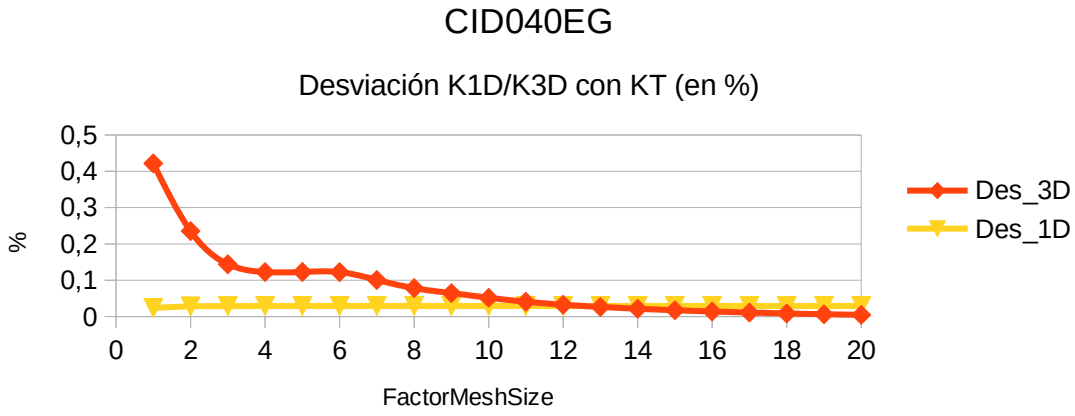

Figura 31:Gráfica comparativa CID040FG. Desviación K3D/K1D vs KT

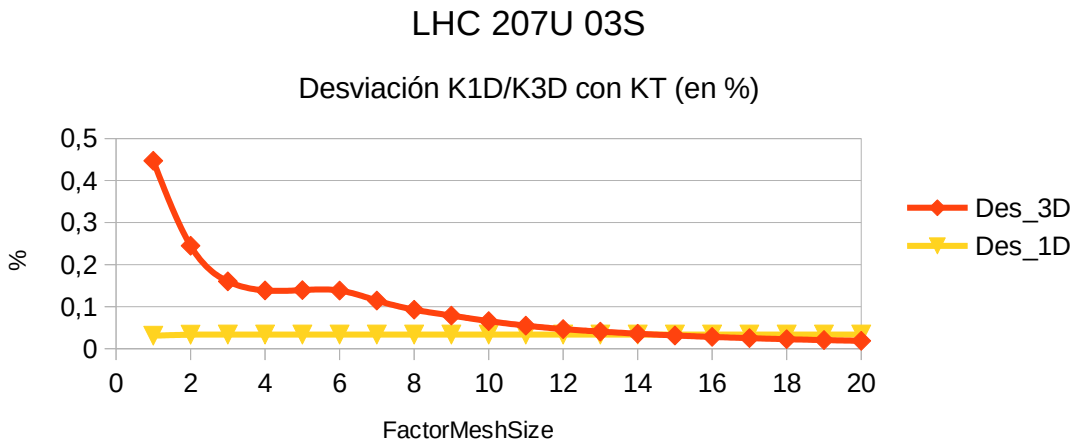

Figura 32: Gráfica comparativa LHC 207U 03S. Desviación K3D/K1D vs KT

En estas gráficas se puede observar como los resultados obtenidos por medio del método en una dimensión tiene una desviación pequeña y que no varía reduciendo el tamaño de malla. También se puede afirmar que con el método en 3D los resultados pueden llegar a tener mas precisión si se reduce lo suficiente el tamaño de malla.

4.3.Discusión de los resultados

En este apartado se discuten los resultados obtenidos a partir de las simulaciones y análisis realizados utilizando diferentes modelos de muelles y tamaños de malla.

En cuanto a los valores de la constante elástica obtenidos, se observa que tanto el modelo 1D como el modelo 3D convergen hacia un valor cercano al valor teórico. El modelo 3D se acerca progresivamente al valor teórico, mientras que el modelo 1D converge hacia el un valor ligeramente inferior al valor teórico. Sin embargo, es importante destacar que el modelo 3D requiere tiempos de cálculo exponencialmente más largos a medida que se incrementa la cantidad de elementos y nodos en la malla, mientras que el modelo 1D mantiene tiempos de cálculo similares para cualquier valor de tamaño de elemento utilizado.

En cuanto a los tiempos totales invertidos, se observa que los tiempos necesarios para resolver el problema mediante el modelo 3D son significativamente mayores a los del modelo 1D debido a la diferencia evidente en el número de elementos utilizados.

En cuanto al análisis del error en los resultados, se observa que el error porcentual disminuye a medida que se reduce el tamaño de malla tanto en el modelo 1D como en el modelo 3D. Sin embargo, el modelo 3D tiende a tener un error porcentual ligeramente menor que el modelo 1D cuando sobrepasamos cierto parámetro de tamaño de elemento.

En general, los resultados obtenidos indican que tanto el modelo 1D como el modelo 3D pueden proporcionar resultados cercanos al valor teórico de la constante elástica del muelle. Sin embargo, el modelo 3D requiere tiempos de

cálculo significativamente más largos. Por lo tanto, la elección del modelo y el tamaño de malla adecuados dependerá de un equilibrio entre la precisión deseada y la eficiencia computacional requerida.

5.Conclusiones

En esta sección, se presentan las conclusiones obtenidas del trabajo realizado, una reflexión crítica sobre el logro de los objetivos planteados inicialmente, un análisis crítico del seguimiento de la planificación y metodología, y las líneas de trabajo futuro que quedaron pendientes de explorar.

Conclusiones del trabajo:

- Se ha desarrollado una metodología para obtener la constante elástica de un muelle helicoidal de sección circular utilizando modelos de elementos finitos en 1D y 3D.
- Se ha implementado un programa en Python para la generación automática de la geometría y mallado utilizando el software Gmsh.
- Se ha utilizado el software Code_Aster para resolver el problema mecánico y obtener los resultados.
- Se ha validado el modelo mediante la comparación de los resultados obtenidos con valores teóricos y con los obtenidos mediante la ley de deformación.
- Se han realizado análisis comparativos de los resultados obtenidos y los tiempos de cálculo utilizando diferentes modelos de muelles y tamaños de malla.

Reflexión sobre el logro de los objetivos planteados:

- En general, se han logrado los objetivos planteados inicialmente. Se ha desarrollado una herramienta que permite obtener la constante elástica de un muelle helicoidal y se ha validado el modelo mediante comparaciones con los valores aportados por los fabricantes de los muelles estudiados y los teóricos obtenidos mediante la ley de deformación.
- Es importante destacar que la precisión y eficiencia computacional del modelo dependen del tamaño de malla utilizado y del enfoque de modelado (1D vs. 3D). Se ha encontrado un equilibrio entre la precisión y la eficiencia en el modelo 1D, mientras que el modelo 3D ofrece resultados más precisos pero requiere tiempos de cálculo significativamente más largos.

Análisis del seguimiento de la planificación y metodología:

- En general, se ha seguido la planificación establecida en términos de los pasos y tareas necesarios para desarrollar la metodología y obtener los resultados.
- Sin embargo, hubo necesidad de introducir cambios en la planificación y metodología. Se realizaron ajustes en el enfoque del trabajo, en el software utilizado y en otros parámetros como el tamaño de malla. Además, se añadió y la funcionalidad de crear modelos computacionales

en 1D y 3D, con el fin de encontrar un equilibrio entre precisión y eficiencia computacional.

Líneas de trabajo futuro pendientes:

- Realizar un análisis más exhaustivo y detallado de diferentes geometrías y propiedades de materiales utilizados en la fabricación de los muelles, con el fin de evaluar la aplicabilidad y precisión del modelo en una amplia gama de situaciones.
- Investigar la influencia de otros factores, como las condiciones de contorno, el tipo de carga aplicada y la no linealidad del material en el comportamiento elástico del muelle.
- Investigar otros parámetros y variables de entrada que puedan influir en la constante elástica del muelle, como la rugosidad de la superficie, la temperatura y la presencia de fatiga o deformación plástica.
- Realizar ensayos de laboratorio para comparar los resultados obtenidos en este trabajo con resultados experimentales.

Para concluir, el trabajo ha logrado desarrollar una metodología para obtener la constante elástica de un muelle helicoidal utilizando modelos de elementos finitos. Se ha validado el modelo y se han obtenido resultados comparativos para diferentes modelos de muelles y tamaños de malla. Sin embargo, existen aspectos adicionales que se pueden explorar en futuras investigaciones para mejorar la precisión, la eficiencia añadir funcionalidad a las herramientas creadas.

6.Glosario

nturns:El número de espiras en el muelle.

dmuelle: El diámetro exterior del muelle.

dalambre: El diámetro del alambre utilizado para el muelle.

hmuelle: La altura del muelle entre las espiras activas.

bturns: El número de espiras en la base del muelle.

E: El límite elástico del material.

NU: El coeficiente de Poisson.

nmuelle: Es el nombre asignado al muelle.

FactorMeshSize: Es un factor de tamaño de malla.

k: constante elástica.

d: diámetro del alambre.

D: diámetro de la espira.

n: número de espiras activas.

G: módulo de cortadura.

FEM: Finite Elements Method.

MEF: Método de los Elementos Finitos.

7.Bibliografía

1. Shimoseki, M., Hamano, T., & Imaizumi, T. (Eds.). (2003). FEM for Springs. Springer.
2. Gamm, H. (1972). El cálculo de muelles helicoidales. Revista CONTRIBUCIONES Científicas y Tecnológicas.
3. Associated Spring. (1987). Engineering Guide to Spring Design. Barnes.
4. Larson, M. G., & Bengzon, F. (2013). The Finite Element Method: Theory, Implementation, and Practice. Springer.
5. Mulla, T., Kadam, S., & Kengar, V. (2012). Finite Element Analysis of Helical Coil Compression Spring for Three Wheeler Automotive Front Suspension. International Journal of Mechanical and Industrial Engineering , volumen(4), Artículo 15.
6. Padron, E. N., Diaz, E. O., & Gonzalez-Carbonell, R. A. (2015). Una introducción al Análisis por Elementos Finitos: aplicaciones y ejemplos. Editorial Universitaria.
7. Wahl, A. M. (1944). Mechanical Spring. Penton Publishing Company.
8. Cellegueta Lizarza, J. T. (2000). Método de los Elementos Finitos para Análisis Estructural. Tecnun.
9. Code_Aster. (n.d.). Recuperado [junio 2023], de http://www.code-aster.org/
10. Gmsh. (n.d.). Recuperado [junio 2023], de http://www.gmsh.info/
11. DIN Deutsches Institut für Normung e. V. (1992). DIN 2098-1: Disc springs, Technical specifications. Berlin, Germany: Beuth Verlag GmbH.

8.Anexos

Anexo I: Extracto catálogo fabricante LeeSpring.

Número de Parte
LHC 207U 03S

Descripción	Especificación
Tipo de Resorte:	Serie Carga Pesada (pulg)
Diámetro exterior (mm):	49.2
Diámetro del orificio (mm):	50.8
Diámetro de poste (mm):	null
Longitud libre (mm):	88.9
Constante (N/mm):	15.084
Altura sólida (mm):	37.59
Diámetro del alambre (mm):	5.26
Material:	Acero inoxidable
Carga a altura sólida (N):	774.38
Extremos:	Escuadrados y rectificados
Dirección de enrollado:	Opción de fábrica
Espiras activas:	5.15
Espiras totales:	7.15
Acabado:	Pasivado según ASTM A967

Figura 33: Oja de datos LHC 207U0 3S. Fuente www.leespring.es.

Número de Parte
CID040EG 02S

Descripción	Especificación
Tipo de Resorte:	Serie DIN-Plus Parte 2
Diámetro exterior (mm):	5.4
Diámetro del orificio (mm):	6
Diámetro de poste (mm):	null
Longitud libre (mm):	16.4
Constante (N/mm):	0.333
Altura sólida (mm):	3.4
Diámetro del alambre (mm):	0.4
Material:	Acero inoxidable
Carga a altura sólida (N):	4.33
Extremos:	Escuadrados
Dirección de enrollado:	Derecho
Espiras activas:	5.5
Espiras totales:	7.5
Acabado:	Pasivado según ASTM A967

Figura 34: Hoja de datos CID040EG 02S. Fuente www.leespring.es.

Aceros Inoxidables								
Material	Especificación comercial disponible	Composición Nominal	Densidad (lb/in3)	Resistencia Tensil Mínima (psi x 10^3)	Módulo de Elasticidad (E)(psi x 10^6)	Módulo de Torsión (G)(psi x 10^6)	Temperatura Máxima de Operación	Descripción y Uso Principal
Acero inoxidable 302/304	ASTM A313	Cr 17.0 - 20.0%, Ni 8.0 - 10.5%	0.29	130 - 325	28	10	550 °F	Alambre para muelle de uso general resistente a la corrosión y al calor. Ligeramente magnético. El Tipo 304 tiene menos carbono que el Tipo 302; sin embargo, es considerado comercialmente equivalente para los muelles de catálogo.
Acero inoxidable 316	ASTM A313	Cr 16.5 - 18.0%, Ni 10.5 - 13.5%, Mo 2.0 - 2.5%	0.29	125 - 245	28	10	550 °F	Alambre para muelle resistente al calor con una mejor resistencia a la corrosión que los de tipo 302/204. Ligeramente magnético. Utilizado a menudo en aplicaciones marítimas, resistente a los ataques del cloruro.

Figura 35: Material muelles LeeSpring. Fuente www.leespring.es.

Anexo I: Extracto catálogo fabricante Lesjöfors.

Material: EN 10270-1-SH								
d_t	D_m	$D_{i\,min}$	L_0	n_t	L_n	F_n	R	Cat.no
0,20	1,32	1,00	5	12,1	3,0	1,4	0,70	1117
14,00	65,0	50,0	100	5,5	83	6725	407	6925

Figura 36: Datos muelles Lesjöfors. Fuente www.lesjoforsab.com.

SS-ref or works ref lengths	Elast. modulus (E)N/mm²	Shearing (G)N/mm²	Density kg/dm³	Size range Wire ∅
EN 10270-1-SH	206 000	81 500	7,85	0,1–12,0

Figura 37: Material muelles Lesjöfors. Fuente www.lesjoforsab.com.

Anexo I: Código del programa kmuelle.py

```
import os
import csv
import time
import math
import kmuelle_funciones

# Get the current working directory
WORKING_DIR = os.getcwd() + '/' + 'tmp'
CURRENT_DIR = os.getcwd()

# Definir la ruta del archivo CSV
archivo_csv = CURRENT_DIR + '/datos.csv'

# Leer el archivo CSV y almacenar las filas actualizadas
filas_actualizadas = []
with open(archivo_csv, 'r') as archivo:
    lector_csv = csv.reader(archivo, delimiter=';')

    # Leer la primera fila y crear las variables
    nvar = next(lector_csv)
    for nombre in nvar:
        globals()[nombre] = None
    indice_CALC = nvar.index('CALC')
    indice_nmuelle = nvar.index('nmuelle')
    indice_KT = nvar.index('KT')
    indice_K3D = nvar.index('K3D')
    indice_K1D = nvar.index('K1D')
    indice_tiempo_gmsh3D = nvar.index('tiempo_gmsh3D')
    indice_tiempo_codeaster3D = nvar.index('tiempo_codeaster3D')
    indice_tiempo_gmsh1D = nvar.index('tiempo_gmsh1D')
    indice_tiempo_codeaster1D = nvar.index('tiempo_codeaster1D')

    # Modificar los valores de K en las filas
```

```
    for fila in lector_csv:
        for i, valor in enumerate(fila):
            variable = nvar[i]
            # Si el valor es un número, convertirlo a float
            try:
                valor = float(valor)
            except ValueError:
                pass

            # Asignar el valor a la variable correspondiente
            globals()[variable] = valor

        # Calcular el valor de K si es igual a 0
        if CALC == 0:
            KT = kmuelle_funciones.KT_calc(E, NU, dmuelle, nturns, dalambre)

                                        K3D,tiempo_gmsh3D,tiempo_codeaster3D    =
kmuelle_funciones.lanzador3D(nturns,        float(dmuelle),        float(dalambre),
float(hmuelle), str(nmuelle), bturns,  FactorMeshSize, E, NU)
                                        K1D,tiempo_gmsh1D,tiempo_codeaster1D    =
kmuelle_funciones.lanzador1D(nturns,        float(dmuelle),        float(dalambre),
float(hmuelle), str(nmuelle), bturns,  FactorMeshSize, E, NU)
        # Actualizar el valor de K en la fila
        fila[indice_CALC] = str(1)
        fila[indice_KT] = str(KT)
        fila[indice_K3D] = str(K3D)
        fila[indice_tiempo_gmsh3D] = str(tiempo_gmsh3D)
        fila[indice_tiempo_codeaster3D] = str(tiempo_codeaster3D)
        fila[indice_K1D] = str(K1D)
        fila[indice_tiempo_gmsh1D] = str(tiempo_gmsh1D)
        fila[indice_tiempo_codeaster1D] = str(tiempo_codeaster1D)

        # Agregar la fila actualizada a la lista de filas actualizadas
        filas_actualizadas.append(fila)

# Sobrescribir el archivo con las filas actualizadas
```

```
with open(archivo_csv, 'w', newline='') as archivo:
    escritor_csv = csv.writer(archivo, delimiter=';')

    # Escribir la primera fila con los nombres de las variables
    escritor_csv.writerow(nvar)

    # Escribir las filas actualizadas en el archivo
    for fila in filas_actualizadas:
        escritor_csv.writerow(fila)
```

Anexo II: Código funciones generadas kmuelle_funciones.py

```
import math
import os
import subprocess
import gmsh
import numpy
import math
import sys
import time
import kmuelle_funciones


def lanzador3D(nturns, dmuelle, dalambre, hmuelle, nmuelle, bturns, FactorMeshSize,E,NU):


    MeshSizeMin = dalambre*1/(FactorMeshSize*3)
    MeshSizeMax = dalambre*1/FactorMeshSize


    WORKING_DIR = os.getcwd() + '/' + 'tmp'
    ASTER_VERSION ='15.2'
    MESH_FILE3D = nmuelle + '3D.unv'
    COMM_FILE3D = nmuelle + '3D.comm'
    EXPORT_FILE3D = nmuelle + '3D.export'


    ASTER_ROOT='/opt/aster'


    if not os.path.exists(WORKING_DIR):
    # El directorio no existe, crearlo
        os.makedirs(WORKING_DIR)


    if not os.path.exists(WORKING_DIR+'/'+nmuelle+'_res3D.txt'):
```

```
        # El archivo no existe, crearlo
            with open(WORKING_DIR+'/'+nmuelle+'_res3D.txt', 'w') as file:
                file.write(" ")


        if not os.path.exists(WORKING_DIR+'/'+COMM_FILE3D):
        # El archivo no existe, crearlo
            with open(WORKING_DIR+'/'+COMM_FILE3D, 'w') as file:
                file.write(" ")


        inicio_gmsh3D = time.time()  # Registra el tiempo de inicio
         nnodos3D = gmsh_springmesh3D(nturns, dmuelle, dalambre, hmuelle,
nmuelle,      WORKING_DIR,      MeshSizeMin,      MeshSizeMax,      bturns,
MESH_FILE3D)
        fin_gmsh3D = time.time()  # Registra el tiempo de finalización
        tiempo_gmsh3D = fin_gmsh3D - inicio_gmsh3D


        fz3D = -1/len(nnodos3D)
        aster_comm3D(WORKING_DIR, COMM_FILE3D, fz3D,E,NU)


        inicio_codeaster3D = time.time()  # Registra el tiempo de inicio
                aster_case3D(nmuelle,    ASTER_VERSION,    MESH_FILE3D,
COMM_FILE3D, EXPORT_FILE3D, ASTER_ROOT, WORKING_DIR)
        fin_codeaster3D = time.time()  # Registra el tiempo de finalización
        # Calcula el tiempo transcurrido
        tiempo_codeaster3D = fin_codeaster3D - inicio_codeaster3D


        archivo=open(WORKING_DIR+'/'+nmuelle+'_res3D.txt','r')


        muelle_res3D = archivo.read()


        if len(muelle_res3D.strip()) == 0:
            K3D = "ERROR"
```

```
            tiempo_gmsh3D = "ERROR"
            tiempo_codeaster3D = "ERROR"
        else:
            lines = muelle_res3D.split('\n')
            # Buscar la columna donde comienza "MAX_DZ"
            header_line = lines[4]
            columns = header_line.split()
            max_dz_column_index = columns.index('MAX_DZ')

            # Obtener el texto de la línea siguiente
            next_line = lines[5]
            next_line_columns = next_line.split()

            # Mostrar la columna y el texto
            K3D=abs(1/float(next_line_columns[max_dz_column_index]))

        return K3D,tiempo_gmsh3D,tiempo_codeaster3D


    def lanzador1D(nturns, dmuelle, dalambre, hmuelle, nmuelle, bturns,
FactorMeshSize,E,NU):

        MeshSizeMin = dalambre*1/(FactorMeshSize*3)
        MeshSizeMax = dalambre*1/FactorMeshSize

        WORKING_DIR = os.getcwd() + '/' + 'tmp'
        ASTER_VERSION='15.2'
        MESH_FILE1D=nmuelle + '1D.unv'
        COMM_FILE1D=nmuelle + '1D.comm'
        EXPORT_FILE1D=nmuelle + '1D.export'

        ASTER_ROOT='/opt/aster'
```

```
        if not os.path.exists(WORKING_DIR):
        # El directorio no existe, crearlo
            os.makedirs(WORKING_DIR)


        if not os.path.exists(WORKING_DIR+'/'+nmuelle+'_res1D.txt'):
        # El archivo no existe, crearlo
            with open(WORKING_DIR+'/'+nmuelle+'_res1D.txt', 'w') as file:
                file.write(" ")


        if not os.path.exists(WORKING_DIR+'/'+COMM_FILE1D):
        # El archivo no existe, crearlo
            with open(WORKING_DIR+'/'+COMM_FILE1D, 'w') as file:
                file.write(" ")


        inicio_gmsh1D = time.time()  # Registra el tiempo de inicio
         nnodos1D = gmsh_springmesh1D(nturns, dmuelle, dalambre, hmuelle, nmuelle, WORKING_DIR, MeshSizeMin, MeshSizeMax, bturns, MESH_FILE1D)


        fin_gmsh1D = time.time()  # Registra el tiempo de finalización
        tiempo_gmsh1D = fin_gmsh1D - inicio_gmsh1D


        fz1D = -1/len(nnodos1D)
              aster_comm1D(WORKING_DIR, COMM_FILE1D, fz1D, E, NU, dalambre)


        inicio_codeaster1D = time.time()  # Registra el tiempo de inicio
                  aster_case1D(nmuelle, ASTER_VERSION, MESH_FILE1D, COMM_FILE1D, EXPORT_FILE1D, ASTER_ROOT, WORKING_DIR)
        fin_codeaster1D = time.time()  # Registra el tiempo de finalización
        # Calcula el tiempo transcurrido
        tiempo_codeaster1D = fin_codeaster1D - inicio_codeaster1D
```

```
        archivo=open(WORKING_DIR+'/'+nmuelle+'_res1D.txt','r')

        muelle_res1D = archivo.read()

        if len(muelle_res1D.strip()) == 0:
            K1D = "ERROR"
            tiempo_gmsh1D = "ERROR"
            tiempo_codeaster1D = "ERROR"
        else:
            lines = muelle_res1D.split('\n')
            # Buscar la columna donde comienza "MAX_DZ"
            header_line = lines[4]
            columns = header_line.split()
            max_dz_column_index = columns.index('MAX_DZ')

            # Obtener el texto de la línea siguiente
            next_line = lines[5]
            next_line_columns = next_line.split()

            # Mostrar la columna y el texto
            K1D=abs(1/float(next_line_columns[max_dz_column_index]))

        return K1D,tiempo_gmsh1D,tiempo_codeaster1D


    def gmsh_springmesh3D(nturns, dmuelle, dalambre, hmuelle, nmuelle,
WORKING_DIR, MeshSizeMin, MeshSizeMax, bturns, MESH_FILE3D):

        # Initialize the Gmsh API
        gmsh.initialize()
        gmsh.clear()

        gmsh.model.add(nmuelle+'3D')
```

```
        # Define parameters
        nptsbase=int(bturns*360*2)
        npts=int(nturns*360*2)
        angbase=bturns*2*math.pi

        wrad = dalambre / 2
        r = (dmuelle - 2 * wrad) / 2
        h = hmuelle
        thetam = 2 * math.pi * nturns * h / (dmuelle - dalambre)
        alfam=math.asin(h/(2*2*r*nturns))
        plb=[]
        psp=[]
        pub=[]

        # Crear puntos de la fijación inferior.
        for j in range(2,nptsbase+1):
            theta = j * angbase / nptsbase
              gmsh.model.occ.addPoint(r * math.cos(theta), -r * math.sin(theta), -
j*1.05/nptsbase*dalambre*angbase/(2*math.pi), 1 ,1000 + nptsbase - j)
            plb.append(1000 + nptsbase - j)

        theta = 0 * angbase / nptsbase
          gmsh.model.occ.addPoint(r * math.cos(theta), -r * math.sin(theta), 0,
1 ,1000 + nptsbase )
        theta = 1 * angbase / nptsbase
            gmsh.model.occ.addPoint(r * math.cos(theta), -r * math.sin(theta), -
1.05/nptsbase*dalambre*angbase/(2*math.pi), 1 ,1000 + nptsbase - 1)

        psp.append(1000 + nptsbase - 2)
        psp.append(1000 + nptsbase - 1)
```

```
        # Crear puntos del muelle.
        psp.append(1000 + nptsbase)
        for i in range(1, npts):
            theta = i * 2 * math.pi * nturns / npts
             gmsh.model.occ.addPoint(r * math.cos(theta), r * math.sin(theta), i *
h / npts, 1 ,1000 + nptsbase + i)
            psp.append(1000 + nptsbase + i)

        psp.append(1000 + nptsbase + npts)

        theta = 2 * math.pi * nturns
          gmsh.model.occ.addPoint(r * math.cos(theta), r * math.sin(theta), h,
1 ,1000 + nptsbase + npts )
        theta = (2 * math.pi * nturns) + 1 * angbase / nptsbase
           gmsh.model.occ.addPoint(r * math.cos(theta), r * math.sin(theta),
h+1.05/nptsbase*dalambre*angbase/(2*math.pi), 1 ,1000 + nptsbase + npts +
1)

        # Crear puntos de la fijación superior.
        for k in range(2,nptsbase+1):
            theta = (2 * math.pi * nturns) + k * angbase / nptsbase
              gmsh.model.occ.addPoint(r * math.cos(theta), r * math.sin(theta),
h+k*1.05/nptsbase*dalambre*angbase/(2*math.pi), 1 ,1000 + nptsbase + npts +
k)
            pub.append(1000 + nptsbase + npts + k)

        psp.append(1000 + nptsbase + npts + 1)
        psp.append(1000 + nptsbase + npts + 2)

        # Create circles
          gmsh.model.occ.addCircle(r * math.cos(2 * math.pi * nturns / 2), r *
math.sin(2 * math.pi * nturns / 2), h / 2, wrad,1010, 0, -math.pi)
        gmsh.model.occ.rotate([(1,1010)],r * math.cos(2 * math.pi * nturns / 2), r
* math.sin(2 * math.pi * nturns / 2), h / 2, 0, 0, 1, 2*math.pi*nturns/2)
```

```
        gmsh.model.occ.rotate([(1,1010)], r * math.cos(2 * math.pi * nturns / 2),
r * math.sin(2 * math.pi * nturns / 2), h / 2, r * math.cos(2 * math.pi * nturns / 2),
r * math.sin(2 * math.pi * nturns / 2), 0, (alfam+math.pi/2))

         gmsh.model.occ.addCircle(r * math.cos(2 * math.pi * nturns / 2), r *
math.sin(2 * math.pi * nturns / 2), h / 2, wrad,1011, -math.pi, 0)

        gmsh.model.occ.rotate([(1, 1011)], r * math.cos(2 * math.pi * nturns / 2),
r * math.sin(2 * math.pi * nturns / 2), h / 2, 0, 0, 1,2 * math.pi * nturns / 2)

        gmsh.model.occ.rotate([(1, 1011)], r * math.cos(2 * math.pi * nturns / 2),
r * math.sin(2 * math.pi * nturns / 2), h / 2,r * math.cos(2 * math.pi * nturns / 2), r
* math.sin(2 * math.pi * nturns / 2), 0,(alfam+math.pi/2))



        gmsh.model.occ.synchronize()
    #    gmsh.fltk.run()

        # Create wires
        gmsh.model.occ.addWire([1011, 1010], 1012)

        # Create plane surface
        gmsh.model.occ.addPlaneSurface([1012], 1003)

        # Create splines
        gmsh.model.occ.addSpline(plb, 1000)
        gmsh.model.occ.addSpline(pub, 1002)
        gmsh.model.occ.addSpline(psp, 1001)

        # Create wires
        gmsh.model.occ.addWire([1000], 1003)
        gmsh.model.occ.addWire([1001], 1004)
        gmsh.model.occ.addWire([1002], 1005)

        gmsh.model.occ.synchronize()

        # Extrude surfaces
```

```
        gmsh.model.occ.addPipe([(2, 1003)], 1004)
        gmsh.model.occ.addPipe([(2, 1004)], 1003)
        gmsh.model.occ.addPipe([(2, 1007)], 1005)

        gmsh.model.occ.synchronize()

        # Launch the GUI to see the results:
        #if '-nopopup' not in sys.argv:
        #    gmsh.fltk.run()

            gmsh.model.occ.fuse([(3,  1)],  [(3,  2),(3,  3)],removeObject  =  True,
removeTool = True)

        gmsh.model.occ.synchronize()

        gmsh.model.addPhysicalGroup(3, [1], 1,"Muelle")
        gmsh.model.addPhysicalGroup(2, [1009, 1007], 2, "Fuerza_r")
        gmsh.model.addPhysicalGroup(2, [1006, 1008], 3, "Fuerza_b")

        gmsh.model.occ.synchronize()

        try :
          threads = max(1,os.cpu_count() - 2)
        except :
          threads = 1

        # Configuración de gmsh
        gmsh.option.setNumber("Mesh.MeshSizeExtendFromBoundary", 0)
        gmsh.option.setNumber("Mesh.MeshSizeFromPoints", 0)
        gmsh.option.setNumber("Mesh.MeshSizeFromCurvature", 0)
        gmsh.option.setNumber("Mesh.MeshSizeMin", MeshSizeMin)
        gmsh.option.setNumber("Mesh.MeshSizeMax", MeshSizeMax)
```

```
        gmsh.option.setNumber("Mesh.MaxNumThreads1D", threads)
        gmsh.option.setNumber("Mesh.MaxNumThreads2D", threads)
        gmsh.option.setNumber("Mesh.MaxNumThreads3D", threads)

        gmsh.option.setNumber('Mesh.Algorithm3D',1)
        gmsh.option.setNumber("Geometry.OCCFixDegenerated", 1)
        gmsh.option.setNumber("Mesh.SaveGroupsOfNodes", 1)

        gmsh.option.setString("Geometry.OCCTargetUnit", 'mm')

        # Generate mesh
        gmsh.model.mesh.generate(3)
        gmsh.model.mesh.optimize('', True)
        gmsh.model.mesh.removeDuplicateNodes()

                                                nnodos3D,          coordnodos          =
gmsh.model.mesh.getNodesForPhysicalGroup(2, 2)

        # Write mesh to UNV format
        unvfile3D = WORKING_DIR + '/' + MESH_FILE3D
        gmsh.write(unvfile3D)

        gmsh.clear()
        # Finalize Gmsh
        gmsh.finalize()

        return nnodos3D

    def gmsh_springmesh1D(nturns, dmuelle, dalambre, hmuelle, nmuelle,
WORKING_DIR, MeshSizeMin, MeshSizeMax, bturns, MESH_FILE1D):

        # Initialize the Gmsh API
```

```
        gmsh.initialize()
        gmsh.clear()

        gmsh.model.add(nmuelle+'1D')

        # Define parameters
        nptsbase=int(bturns*360*2)
        npts=int(nturns*360*2)
        angbase=bturns*2*math.pi

        MeshSizeMax1D=MeshSizeMax
        if MeshSizeMax1D < dalambre/6:
            MeshSizeMax1D = dalambre/6

        wrad = dalambre / 2
        r = (dmuelle - 2 * wrad) / 2
        h = hmuelle
        thetam = 2 * math.pi * nturns * h / (dmuelle - dalambre)
        alfam=math.asin(h/(2*2*r*nturns))
        plb=[]
        psp=[]
        pub=[]

        # Crear puntos de la fijación inferior.
        for j in range(2,nptsbase+1):
            theta = j * angbase / nptsbase
              gmsh.model.occ.addPoint(r * math.cos(theta), -r * math.sin(theta), -
j/nptsbase*dalambre*angbase/(2*math.pi), 1 ,1000 + nptsbase - j)
            plb.append(1000 + nptsbase - j)

        theta = 0 * angbase / nptsbase
```

```
            gmsh.model.occ.addPoint(r * math.cos(theta), -r * math.sin(theta), 0,
1 ,1000 + nptsbase )

        theta = 1 * angbase / nptsbase

            gmsh.model.occ.addPoint(r * math.cos(theta), -r * math.sin(theta), -
1/nptsbase*dalambre*angbase/(2*math.pi), 1 ,1000 + nptsbase - 1)


        psp.append(1000 + nptsbase - 2)

        psp.append(1000 + nptsbase - 1)


        # Crear puntos del muelle.

        psp.append(1000 + nptsbase)

        for i in range(1, npts):

            theta = i * 2 * math.pi * nturns / npts

             gmsh.model.occ.addPoint(r * math.cos(theta), r * math.sin(theta), i *
h / npts, 1 ,1000 + nptsbase + i)

            psp.append(1000 + nptsbase + i)


        psp.append(1000 + nptsbase + npts)


        theta = 2 * math.pi * nturns

           gmsh.model.occ.addPoint(r * math.cos(theta), r * math.sin(theta), h,
1 ,1000 + nptsbase + npts )

        theta = (2 * math.pi * nturns) + 1 * angbase / nptsbase

             gmsh.model.occ.addPoint(r * math.cos(theta), r * math.sin(theta),
h+1/nptsbase*dalambre*angbase/(2*math.pi), 1 ,1000 + nptsbase + npts + 1)


        # Crear puntos de la fijación superior.

        for k in range(2,nptsbase+1):

            theta = (2 * math.pi * nturns) + k * angbase / nptsbase

                gmsh.model.occ.addPoint(r * math.cos(theta), r * math.sin(theta),
h+k/nptsbase*dalambre*angbase/(2*math.pi), 1 ,1000 + nptsbase + npts + k)

            pub.append(1000 + nptsbase + npts + k)


        psp.append(1000 + nptsbase + npts + 1)
```

```
psp.append(1000 + nptsbase + npts + 2)

gmsh.model.occ.synchronize()
#gmsh.fltk.run()

# Create splines
gmsh.model.occ.addSpline(plb, 1000)
gmsh.model.occ.addSpline(pub, 1002)
gmsh.model.occ.addSpline(psp, 1001)
gmsh.model.occ.synchronize()

gmsh.model.occ.synchronize()

gmsh.model.addPhysicalGroup(1, [1000], 2, "Fuerza_b")
gmsh.model.addPhysicalGroup(1, [1002], 3, "Fuerza_r")
gmsh.model.addPhysicalGroup(1, [1001], 1, "Muelle")

try :
  threads = max(1,os.cpu_count() - 2)
except :
  threads = 1

# Configuración de gmsh
gmsh.option.setNumber("Mesh.MeshSizeMin", MeshSizeMin)
gmsh.option.setNumber("Mesh.MeshSizeMax", MeshSizeMax1D)
gmsh.option.setNumber("Mesh.MaxNumThreads1D", threads)
gmsh.option.setNumber("Mesh.MaxNumThreads1D", threads)
gmsh.option.setNumber("Mesh.MaxNumThreads3D", threads)

gmsh.option.setNumber('Mesh.Algorithm3D',1)
gmsh.option.setNumber("Geometry.OCCFixDegenerated", 1)
```

```
        gmsh.option.setNumber("Mesh.SaveGroupsOfNodes", 1)

        gmsh.option.setString("Geometry.OCCTargetUnit", 'mm')

        # Generate mesh
        gmsh.model.mesh.generate(1)
        gmsh.model.mesh.optimize('', True)
        gmsh.model.mesh.removeDuplicateNodes()

                                                nnodos1D,          coordnodos          =
gmsh.model.mesh.getNodesForPhysicalGroup(1, 3)

        # Write mesh to UNV format
        unvfile1D = WORKING_DIR + '/' + MESH_FILE1D
        gmsh.write(unvfile1D)

        gmsh.clear()
        # Finalize Gmsh
        gmsh.finalize()

        return nnodos1D

    def    aster_case3D(nmuelle,    ASTER_VERSION,    MESH_FILE3D,
COMM_FILE3D, EXPORT_FILE3D, ASTER_ROOT, WORKING_DIR):
        try :
          threads = max(1,os.cpu_count() - 2)
        except :
          threads = 1

        exportfile = os.path.join(WORKING_DIR,EXPORT_FILE3D)
        e = open(exportfile,'w')
        e.write("P actions make_etude"+'\n')
```

```
        e.write("P mem_aster 80.0"+'\n')
        e.write("P memjob 33554432"+'\n')
        e.write("P testlist verification sequential"+'\n')
        e.write("P mode interactive"+'\n')
        e.write("P mpi_nbcpu 1"+'\n')
        e.write("P mpi_nbnoeud 1"+'\n')
        e.write("P ncpus 1"+'\n')
        e.write("P platform Linux64"+'\n')
        e.write("P proxy_dir " + WORKING_DIR +'\n')
        e.write("P rep_trav " + WORKING_DIR + "/case_aster"+'\n')
        e.write("P version " + ASTER_VERSION + '\n')
        e.write("A args "+'\n')
        e.write("A memory 32768.0"+'\n')
        e.write("A tpmax 9000.0"+'\n')
         e.write("F comm " + WORKING_DIR + '/' + COMM_FILE3D +" D  1"+'\
n')
        e.write("F libr " + WORKING_DIR + '/' + MESH_FILE3D + " D  20"+'\n')
        e.write("F libr " + WORKING_DIR + '/' + nmuelle + "_res3D.txt R  8"+'\n')
          e.write("F  libr  "  +  WORKING_DIR  +  '/'  +  nmuelle  +  "_res3D.med  R
80"+'\n')
         e.write("F mess " + WORKING_DIR + '/' + nmuelle + "3D.message R
6"+'\n')
        e.close()


          run=(ASTER_ROOT  +  '/bin/codeaster-run  '  +  WORKING_DIR  +  '/'  +
EXPORT_FILE3D)


        subprocess.run(run,shell=True)


    def     aster_case1D(nmuelle,     ASTER_VERSION,     MESH_FILE1D,
COMM_FILE1D, EXPORT_FILE1D, ASTER_ROOT, WORKING_DIR):
        try :
          threads = max(1,os.cpu_count() - 2)
```

```
        except :
          threads = 1
        exportfile = os.path.join(WORKING_DIR,EXPORT_FILE1D)
        e = open(exportfile,'w')
        e.write("P actions make_etude"+'\n')
        e.write("P mem_aster 80.0"+'\n')
        e.write("P memjob 33554432"+'\n')
        e.write("P testlist verification sequential"+'\n')
        e.write("P mode interactive"+'\n')
        e.write("P mpi_nbcpu 1"+'\n')
        e.write("P mpi_nbnoeud 1"+'\n')
        e.write("P ncpus 1"+'\n')
        e.write("P platform Linux64"+'\n')
        e.write("P proxy_dir " + WORKING_DIR +'\n')
        e.write("P rep_trav " + WORKING_DIR + "/case_aster"+'\n')
        e.write("P version " + ASTER_VERSION + '\n')
        e.write("A args "+'\n')
        e.write("A memory 32768.0"+'\n')
        e.write("A tpmax 9000.0"+'\n')
         e.write("F comm " + WORKING_DIR + '/' + COMM_FILE1D +" D  1"+'\
n')
        e.write("F libr " + WORKING_DIR + '/' + MESH_FILE1D + " D  20"+'\n')
        e.write("F libr " + WORKING_DIR + '/' + nmuelle + "_res1D.txt R  8"+'\n')
          e.write("F  libr  "  +  WORKING_DIR  +  '/'  +  nmuelle  +  "_res1D.med  R
80"+'\n')
         e.write("F mess " + WORKING_DIR + '/' + nmuelle + "1D.message R
6"+'\n')
        e.close()


          run=(ASTER_ROOT  +  '/bin/codeaster-run  '  +  WORKING_DIR  +  '/'  +
EXPORT_FILE1D)


        subprocess.run(run,shell=True)
```

```
def aster_comm3D(WORKING_DIR, COMM_FILE3D, fz3D,E,NU):

    commfile3D = os.path.join(WORKING_DIR,COMM_FILE3D)
    c = open(commfile3D,'w')

    c.write("DEBUT(LANG='EN')"+'\n')
    c.write(" "+'\n')
    c.write(" "+'\n')
    c.write("mesh = LIRE_MAILLAGE(FORMAT='IDEAS',"+'\n')
    c.write("                     UNITE=20)"+'\n')
    c.write(" "+'\n')
                                                    c.write("model                =
AFFE_MODELE(AFFE=_F(MODELISATION=('3D', ),"+'\n')
    c.write("                         PHENOMENE='MECANIQUE',"+'\n')
    c.write("                         TOUT='OUI'),"+'\n')
    c.write("                     MAILLAGE=mesh)"+'\n')
    c.write(" "+'\n')
    c.write("mater = DEFI_MATERIAU(ELAS=_F(E="+str(E)+","+'\n')
    c.write("                           NU="+str(NU)+"))"+'\n')
    c.write(" "+'\n')
     c.write("fieldmat = AFFE_MATERIAU(AFFE=_F(MATER=(mater, ),"+'\
n')
    c.write("                           TOUT='OUI'),"+'\n')
    c.write("                       MAILLAGE=mesh)"+'\n')
    c.write(" "+'\n')
    c.write("load = AFFE_CHAR_MECA(DDL_IMPO=_F(DX=0.0,"+'\n')
    c.write("                           DY=0.0,"+'\n')
    c.write("                           GROUP_MA=('Fuerza_r', )),"+'\n')
    c.write("                     MODELE=model)"+'\n')
    c.write(" "+'\n')
    c.write("load0 = AFFE_CHAR_MECA(DDL_IMPO=_F(DX=0.0,"+'\n')
```

```
        c.write("                       DY=0.0,"+'\n')
        c.write("                       DZ=0.0,"+'\n')
        c.write("                       GROUP_MA=('Fuerza_b', )),"+'\n')
        c.write("                MODELE=model)"+'\n')
        c.write(" "+'\n')
                                                    c.write("load1          =
AFFE_CHAR_MECA(FORCE_NODALE=_F(FZ="+str(fz3D)+','+'\n')
        c.write("                       GROUP_NO=('Fuerza_r', )),"+'\n')
        c.write("                MODELE=model)"+'\n')
        c.write(" "+'\n')
    #                                               c.write("load2          =
AFFE_CHAR_MECA(LIAISON_SOLIDE=_F(GROUP_NO=('Fuerza_r', )),"+'\n')
    #   c.write("                MODELE=model)"+'\n')
    #   c.write(" "+'\n')
        c.write("reslin = MECA_STATIQUE(CHAM_MATER=fieldmat,"+'\n')
        c.write("                EXCIT=(_F(CHARGE=load),"+'\n')
        c.write("                       _F(CHARGE=load0),"+'\n')
    #   c.write("                       _F(CHARGE=load2),"+'\n')
        c.write("                       _F(CHARGE=load1)),"+'\n')
        c.write("                MODELE=model,"+'\n')
        c.write("                SOLVEUR=_F(RESI_RELA=0.0001))"+'\n')
        c.write(" "+'\n')
                                                    c.write("table6         =
POST_ELEM(MINMAX=_F(GROUP_MA=('Fuerza_r', ),"+'\n')
        c.write("                       NOM_CHAM='DEPL',"+'\n')
        c.write("                       NOM_CMP=('DZ', ),"+'\n')
        c.write("                       RESULTAT=reslin))"+'\n')
        c.write(" "+'\n')
        c.write("IMPR_RESU(FORMAT='MED',"+'\n')
        c.write("       RESU=_F(NOM_CHAM=('DEPL', ),"+'\n')
        c.write("               RESULTAT=reslin),"+'\n')
        c.write("       UNITE=80)"+'\n')
```

```
        c.write(" "+'\n')
        c.write("IMPR_TABLE(TABLE=table6,"+'\n')
        c.write("          UNITE=8)"+'\n')
        c.write(" "+'\n')
        c.write("FIN()"+'\n')


    def aster_comm1D(WORKING_DIR, COMM_FILE1D,
fz1D,E,NU,dalambre):
        ralambre=dalambre/2
        commfile1D = os.path.join(WORKING_DIR,COMM_FILE1D)
        c = open(commfile1D,'w')


        c.write("DEBUT()"+'\n')
        c.write(" "+'\n')
        c.write(" "+'\n')
        c.write("mesh = LIRE_MAILLAGE(FORMAT='IDEAS',"+'\n')
        c.write("                UNITE=20)"+'\n')
        c.write(" "+'\n')
                                                c.write("model                =
AFFE_MODELE(AFFE=_F(MODELISATION=('POU_D_T', ),"+'\n')
        c.write("                    PHENOMENE='MECANIQUE',"+'\n')
        c.write("                        TOUT='OUI'),"+'\n')
        c.write("                    MAILLAGE=mesh)"+'\n')
        c.write(" "+'\n')
        c.write("elemprop = AFFE_CARA_ELEM(MODELE=model,"+'\n')
        c.write("                  POUTRE=_F(CARA=('R',),"+'\n')
         c.write("                              GROUP_MA=('Fuerza_r', 'Fuerza_b',
'Muelle'),"+'\n')
        c.write("                    SECTION='CERCLE',"+'\n')
        c.write("                    VALE=("+str(ralambre)+",)))"+'\n')
        c.write(" "+'\n')
        c.write("mater = DEFI_MATERIAU(ELAS=_F(E="+str(E)+","+'\n')
```

```
        c.write("                        NU="+str(NU)+"))"+'\n')
        c.write(" "+'\n')
         c.write("fieldmat = AFFE_MATERIAU(AFFE=_F(MATER=(mater, ),"+'\
n')
        c.write("                        TOUT='OUI'),"+'\n')
        c.write("                    MAILLAGE=mesh)"+'\n')
        c.write(" "+'\n')
        c.write("load = AFFE_CHAR_MECA(DDL_IMPO=_F(DX=0.0,"+'\n')
        c.write("                        DY=0.0,"+'\n')
        c.write("                        DRX=0.0,"+'\n')
        c.write("                        DRY=0.0,"+'\n')
        c.write("                        DRZ=0.0,"+'\n')
        c.write("                        GROUP_MA=('Fuerza_r', )),"+'\n')
        c.write("                MODELE=model)"+'\n')
        c.write(" "+'\n')
        c.write("load0 = AFFE_CHAR_MECA(DDL_IMPO=_F(DX=0.0,"+'\n')
        c.write("                        DY=0.0,"+'\n')
        c.write("                        DZ=0.0,"+'\n')
        c.write("                        GROUP_NO=('Fuerza_b', )),"+'\n')
        c.write("                MODELE=model)"+'\n')
        c.write(" "+'\n')
                                                c.write("load1              =
AFFE_CHAR_MECA(FORCE_NODALE=_F(FZ="+str(fz1D)+','+'\n')
        c.write("                        GROUP_NO=('Fuerza_r', )),"+'\n')
        c.write("                MODELE=model)"+'\n')
        c.write(" "+'\n')
    #                                           c.write("load2              =
AFFE_CHAR_MECA(LIAISON_SOLIDE=_F(GROUP_NO=('Fuerza_r', )),"+'\n')
    #   c.write("                MODELE=model)"+'\n')
    #   c.write(" "+'\n')
        c.write("reslin = MECA_STATIQUE(CARA_ELEM=elemprop,"+'\n')
        c.write("                CHAM_MATER=fieldmat,"+'\n')
```

```
        c.write("                    EXCIT=(_F(CHARGE=load),"+'\n')
        c.write("                          _F(CHARGE=load0),"+'\n')
    #    c.write("                          _F(CHARGE=load2)),"+'\n')
        c.write("                          _F(CHARGE=load1)),"+'\n')
        c.write("                    MODELE=model,"+'\n')
        c.write("                    SOLVEUR=_F(RESI_RELA=0.0001))"+'\n')
        c.write(" "+'\n')
                                                        c.write("table6                =
POST_ELEM(MINMAX=_F(GROUP_MA=('Fuerza_r', ),"+'\n')
        c.write("                          NOM_CHAM='DEPL',"+'\n')
        c.write("                          NOM_CMP=('DZ', ),"+'\n')
        c.write("                          RESULTAT=reslin))"+'\n')
        c.write(" "+'\n')
        c.write("IMPR_RESU(FORMAT='MED',"+'\n')
        c.write("          RESU=_F(NOM_CHAM=('DEPL', ),"+'\n')
        c.write("                  RESULTAT=reslin),"+'\n')
        c.write("          UNITE=80)"+'\n')
        c.write(" "+'\n')
        c.write("IMPR_TABLE(TABLE=table6,"+'\n')
        c.write("           UNITE=8)"+'\n')
        c.write(" "+'\n')
        c.write("FIN()"+'\n')


    def KT_calc(E, NU, dmuelle, nturns, dalambre):
        G = E/(2*(1 + NU))
        KT=(G*(dalambre)**4)/(8*(dmuelle-dalambre)**3*nturns)
        return KT
```